Centro Universitário de Brasília
Instituto CEUB de Pesquisa e Desenvolvimento – ICPD

FELIPE RODRIGUES DE VASCONCELLOS

AUDITORIAS DE SEGURANÇA: BENEFÍCIOS DA AUTOMAÇÃO EM DETRIMENTO DO FATOR HUMANO

Brasília
2016

FELIPE RODRIGUES DE VASCONCELLOS

AUDITORIAS DE SEGURANÇA: BENEFÍCIOS DA AUTOMAÇÃO EM DETRIMENTO DO FATOR HUMANO

Trabalho apresentado ao Centro Universitário de Brasília (UniCEUB/ICPD) como uma das atividades do programa de Metodologia Científica do curso de Pós-Graduação Lato Sensu na área de Redes com ênfase em Segurança.
Orientador: Prof. MsC Francisco Javier De Obaldía Díaz

FELIPE RODRIGUES DE VASCONCELLOS
AUDITORIAS DE SEGURANÇA: BENEFÍCIOS DA AUTOMAÇÃO EM DETRIMENTO DO FATOR HUMANO

Trabalho apresentado ao Centro Universitário de Brasília (UniCEUB/ICPD) como uma das atividades do programa de Metodologia Científica do curso de Pós-Graduação Lato Sensu na área de Redes com ênfase em Segurança.
Orientador: Prof. MsC Francisco Javier De Obaldía Díaz

Brasília, ___ de _____ de 2016.

Banca Examinadora

_____

Prof. MsC Francisco Javier de Obaldía Díaz

_____

Prof. Dr. Gilson Ciarallo

_____

Prof. MsC Marco Antônio de Oliveira Araújo

AGRADECIMENTOS

Agradeço ao meu orientador, professor MsC Francisco Javier de Obaldía Díaz, pelos conselhos oportunos e corretas orientações. Agradeço ainda aos profissionais que participaram dos testes realizados durante à pesquisa e à minha esposa que esteve ao meu lado durante todo o “percurso”.

RESUMO

Este trabalho tem como objetivo analisar as vantagens do uso de uma ferramenta automatizada de *pentest* em atividades de auditoria de segurança. Para alcançar o objetivo julgou-se necessário desenvolver uma ferramenta que atendesse as necessidades dos profissionais de segurança da informação. Destarte, foi distribuído um questionário para esse público que auxiliou na definição das funcionalidades necessários ao novo software. Em seguida, foi realizada uma análise das principais metodologias de auditoria de segurança com reconhecimento internacional e eleita uma destas para nortear o desenvolvimento do software. Com base nos requisitos levantados e na metodologia selecionada foi desenvolvida a ferramenta (SAPINES) em Shell Script e Python para tornar as tarefas de um teste de penetração mais céleres e metodológicas, garantindo uma eficiência mais refinada através da minimização do fator humano nas ações de descoberta e exploração de vulnerabilidades em sistemas informáticos. Por fim, foi criado um cenário de testes onde três pares de voluntários com conhecimentos heterogêneos em segurança da informação realizaram uma auditoria de segurança com e sem o uso da ferramenta desenvolvida, com vistas a mensurar as eventuais vantagens e prejuízos de seu uso. Com base nos resultados concluiu-se que o uso de softwares automatizados melhora o rendimento dos profissionais de segurança em ações de *pentest*.

Palavras-chave: Sapiens. Pentest. Auditoria de segurança. Python. Shell script.

RESUMEN

Este trabajo tiene el objetivo de analizar las ventajas de utilizar una herramienta automatizada de pentest en las actividades de auditoría de seguridad. Para lograr el objetivo se consideró necesario desarrollar una herramienta que satisfaga las necesidades de los profesionales de la seguridad de la información. De este modo, se distribuyó un cuestionario para este público que ayudaba a definir las características necesarias para un nuevo software. A continuación, se realizó un análisis de las principales metodologias de auditoría de seguridad con reconocimiento internacional y eligió uno de estos para guiar el desarrollo de software. Sobre la base de los requisitos planteados y la metodología seleccionada se desarrolló la herramienta (SAPINES) con Python y Shell para hacer las tareas de una prueba más rápida y metodológico de penetración, lo que garantiza un rendimiento más refinado, reduciendo al mínimo el factor humano en las acciones de descubrimiento y aprovecha vulnerabilidades en los sistemas informáticos. Por último, se creó un escenario de prueba en el que tres pares de voluntarios con conocimientos heterogéneos en seguridad de la información llevó a cabo una auditoría de seguridad con y sin el uso de herramientas desarrolladas con el fin de medir los beneficios y daños potenciales de su uso. Basándose en los resultados, se concluyó que el uso de software automatizado mejora el rendimiento de los profesionales de la seguridad em las acciones PenTest.

Palabras clave: Sapiens. Pentest. Auditoría de seguridad. Python. Shell script.

SUMÁRIO

INTRODUÇÃO

A inserção dos dispositivos computacionais na sociedade contemporânea transformou as relações humanas, conferindo inúmeras facilidades e novas formas de interações econômicas e interpessoais, como as compras on-line e as mídias sociais. Essa escalada tecnológica trouxe, acompanhada dos benefícios, novos desafios referentes a segurança da informação e comunicações.

Diante da necessidade urgente de inserção nessa nova realidade, indivíduos e organizações demandam cada vez mais dispositivos computacionais para realizar atividades pessoais e/ou econômicas, multiplicando e capilarizando a presença de computadores, servidores, smartphones e outros equipamentos digitais nas atividades cotidianas. Acompanhando o aumento do uso da tecnologia, surgiram novas abordagens para antigos crimes, que passaram a ser efetivados sobre uma nova plataforma, o computador.

Nesse cenário, as empresas tornaram-se alvos compensadores dos crimes cibernéticos (realizados através de dispositivos computacionais) devido a relevância e volume dos dados que armazenam e pelas operações financeiras que algumas realizam. Ilustra esse fato o relatório apresentado em 2015 pela consultoria Grant Thornton, que ouviu mais de 2.5 mil executivos em 35 países e concluiu que mais de uma em cada seis empresas foi alvo de algum ataque cibernético em 2014, acarretando um prejuízo estimado em US$ 315 bilhões. [1]

Para minimizar os efeitos do arsenal tecnológico empregado contra seus ativos de tecnologia da informação, as organizações buscam mitigar as eventuais vulnerabilidades existentes em sua infraestrutura implementando sistemas de segurança em suas redes, como: *firewalls, Intrusion Detection System* (IDS) e *Security Information and Event Management* (SIEM). Todavia, a eficácia destes equipamentos não é absoluta e a equação da segurança não é resolvida com a simples adição de inúmeros dispositivos como estes. A interação entre inúmeros softwares e hardwares de diferentes fabricantes e o caráter dinâmico da tecnologia, tornam as redes de computadores um ambiente extremamente propenso a brechas de segurança e bastante suscetível a fértil imaginação dos atacantes.

[1] Disponível em: http://convergenciadigital.uol.com.br/cgi/cgilua.exe/sys/start.htm?infoid=40905&sid=18 Acesso em: 17 dez. 2015.

Uma forma de dirimir ações hostis em redes de computadores, além dos já citados equipamentos de segurança, é a realização de testes de penetração, que é definido pelo *National Institute of Standards and Technology* (2008) da seguinte maneira:

> Penetration testing is security testing in which assessors mimic real-world attacks to identify methods for circumventing the security features of an application, system, or network. It often involves launching real attacks on real systems and data that use tools and techniques commonly used by attackers. Most penetration tests involve looking for combinations of vulnerabilities on one or more systems that can be used to gain more access than could be achieved through a single vulnerability. Penetration testing can also be useful for determining:
> - How well the system tolerates real world-style attack patterns;
> - The likely level of sophistication an attacker needs to successfully compromise the system;
> - Additional countermeasures that could mitigate threats against the system;
> - Defenders' ability to detect attacks and respond appropriately;
>
> Penetration testing can be invaluable, but it is labor-intensive and requires great expertise to minimize the risk to targeted systems. Systems may be damaged or otherwise rendered inoperable during the course of penetration testing, even though the organization benefits in knowing how a system could be rendered inoperable by an intruder. Although experienced penetration testers can mitigate this risk, it can never be fully eliminated. Penetration testing should be performed only after careful consideration, notification.

Chama atenção na definição apresentada pelo NIST o caráter realístico desse tipo de atividade, que lança mão de ataques reais na tentativa de encontrar vulnerabilidades que podem ser exploradas por indivíduos mal intencionados. Por conta das características que cercam a atividade, o Instituto destaca a necessidade de "*great expertise*" para realizar um *pentest*, pois os riscos de danos aos sistemas alvo são reais e devem minimizados.

Outra definição de pentest é apresentada pela *Crisalis Securing Critical Infrastructures* (2011):

> Per definition, a penetration test is a simulation of an attack aimed at assessing the security of a computer system or a network. Simulation refers to the fact that the test is conducted by the system owner or on his request. This test has multiple phases and allows to highlight infrastructures' and protocols' weaknesses. During a penetration test, the penetration tester tries to discover and sometimes exploit known or unknown vulnerabilities in order to obtain useful information or to access targeted hosts. At the end, all security problems discovered are reported to the system owner together with an assessment of their impact on the system. The analysis should also propose technical or
> organizational mitigations to identified problems.

A definição acima foca nas diferentes etapas do *pentest*, passando desde a identificação de vulnerabilidades conhecidas e/ou desconhecidas até a entrega de um relatório ao proprietário do sistema alvo apontando as falhas e sugerindo correções. Com essa definição fica evidente que o objetivo final de um teste de penetração é avaliar a segurança em uma rede de computadores ou dispositivos computacionais, com vistas a restringir os riscos provocados por ameaças internas e externas.

Para alcançar o escopo citado é indicado seguir uma abordagem metodológica para guiar as ações de maneira organizada e sistemática, evitando assim que alguma falha não seja descoberta e permaneça disponível para ser utilizada por indivíduos pouco éticos.
Diante das incontáveis portas lógicas, aplicações, códigos, serviços e protocolos em uma rede de computadores, é indiscutível o caráter técnico de um pentest. Dessa forma, a realização dessa medida profilática, em regra, é executada apenas por profissionais com experiência na área, evitando, dessa maneira, ações danosas e definitivas aos sistemas alvo.

Diante do exposto, torna-se evidente que a qualidade e eficácia de um teste de penetração depende da qualidade e conhecimento do profissional executante. Contudo, dominar todas as particularidades de um complexo sistema computacional é algo improvável, e fatalmente inúmeras brechas deixariam de ser observadas, não apenas pela falta de conhecimento do *pentester*, mas também pelo grande fluxo de informações que são analisados em atividades dessa natureza.

Para minimizar essas limitações, as ferramentas automatizadas de *pentest* surgem como uma solução interessante, pois executam a tarefa que seria realizada "manualmente" por um indivíduo de maneira completa e independente, sem as restrições impostas pelo conhecimento do profissional que executa o teste. No entanto, cabe destacar que tais ferramentas não são totalmente autossuficientes e dependem de parâmetros e ajustes, além da interpretação dos dados obtidos.

Atualmente existem alguns softwares comerciais com esse propósito, como o Core Impact, o Metasploit e o Canvas. Nenhuma dessas soluções são proprietárias de empresas nacionais e devido ao elevado poder estratégico a importação para aquisição por órgãos governamentais brasileiros é controlada. Outro aspecto relevante é o elevado valor cobrado para aquisição das soluções citadas, podendo chegar a US 5.000 o valor da licença, como o Metasploit Express[2]. Por fim, devido ao caráter comercial adotado pelos seus desenvolvedores, o código de nenhuma das três ferramentas é aberto, o que inviabiliza uma análise apurada do que efetivamente ocorre quando o software é executado, imprimindo um fator crítico ao seu uso, pois dependendo da criticidade e sigilo da rede auditada, o desconhecimento do código fonte torna-se um fator impeditivo do seu uso, como em infraestruturas críticas e redes militares, por exemplo.

Diante do exposto, ficam evidentes os benefícios que uma ferramenta de automação de *pentest* traz consigo. Sendo uma solução que depende de pouca interação com o usuário, a

[2] Disponível em: https://www.rapid7.com/store/. Acesso em 11 fev 2016.

solução realiza verificações que podem ir além dos conhecimentos dos profissionais que conduzem o *pentest*, garantindo testes mais completos e detalhados, além de tornar a tarefa muito mais célere e eficiente, pois segue uma trilha metodológica previamente imposta em seu código. Frente às soluções de mercado supracitadas, a ferramenta proposta nesse trabalho tem a clara vantagem de ser mais flexível e personalizável, pois seu código não é proprietário.

Com base nas informações apresentadas surge o seguinte problema a ser pesquisado: a automação das etapas de um teste de penetração garante maior eficiência e eficácia ao ofício do *pentester*?

Para responder o problema apresentado, algumas hipóteses podem ser sugeridas:

H1: A utilização de uma ferramenta automatizada torna as ações de um teste de penetração mais rápidas devido ao maior poder de processamento do software em comparação com os *pentesters*.

H2: A adoção de uma ferramenta automatizada não garante maior celeridade ao teste de penetração, pois são processadas inúmeras possibilidades de ataque, que demandam recurso computacional e tempo que seriam minimizados caso um *pentester* participasse ativamente do teste, canalizando o esforço em atividades identificadas por ele como relevantes.

H3: O uso de uma ferramenta automatizada impõe uma série de testes pré-definidos a serem executados, garantindo uma cobertura mais completa, quando comparado com a ação de um profissional de segurança da informação, de todas as eventuais vulnerabilidades presentes em uma rede alvo. Dessa maneira, eficácia de testes que lançam mão de softwares dessa natureza é maior do que os demais.

H4: O fator humano é decisivo em um *pentest* e a experiência do profissional que executa essa tarefa é determinante para o seu sucesso. Priorizar o uso de uma ferramenta em detrimento da inciativa de um profissional limita suas ações e prejudica o resultado.

Diante disso, os testes conduzidos sem as ferramentas de automação são mais eficazes, pois não cinge a “pro atividade” dos *pentesters.*

Com base no problema levantando e nas hipóteses sugeridas, o objetivo geral da presente pesquisa é analisar as vantagens do uso de uma ferramenta automatizada de *pentest* em atividades de auditoria de segurança.

Para alcançar com êxito o escopo principal, sugere-se os seguintes objetivos específicos a serem alcançados:

1. Definir uma metodologia de teste de penetração efetiva para ser adotada como referência no desenvolvimento da ferramenta;
2. Definir os requisitos desejáveis para uma ferramenta de *pentest*;

3. Definir um laboratório de testes para avaliar a eficácia/eficiência da ferramenta;
4. Desenvolver uma ferramenta para tornar as tarefas de um teste de penetração mais céleres e metodológicas, garantindo uma eficiência mais refinada através da minimização do fator humano nas ações de descoberta e exploração de vulnerabilidades em sistemas informáticos. Tal ferramenta será utilizada durante os testes conduzidos entre profissionais que a utilizarão e outros que realizarão a auditoria contando apenas com seus conhecimentos, como será detalhado no capítulo referente a metodologia.

Para alcançar os objetivos listados a presente pesquisa está organizada em 5 capítulos além deste introdutório. O capítulo 1 apresentará o referencial teórico necessário para a total compreensão da ferramenta a ser desenvolvida, apresentando as metodologias de pentest, as características das linguagens de programação e outros *softwares* utilizadas durante a pesquisa. O capítulo 2 trará os aspectos metodológicas utilizados, detalhando as etapas percorridas para alcançar o produto final. No terceiro capítulo a ferramenta desenvolvida e batizada de SAPIENS será apresentada, bem como os resultados obtidos nos testes conduzidos.

A partir deste ponto da pesquisa, as menções à **ferramenta** criada podem ser feitas através do substantivo comum em negrito ou por meio do substantivo próprio SAPIENS. Ambas as formas serão empregadas devendo ser entendidas como referência ao código em Python e Shell Script produzido por este autor e apresentado nas páginas a seguir.

1 REFERENCIA TEÓRICO

O incremento do uso computacional nas mais variadas atividades humanas fomentou o desenvolvimento da segurança da informação com vistas a minimizar o impacto de eventuais ações maliciosas contra as redes de computadores.

Existem algumas medidas que podem ser adotadas para proteger redes lógicas, uma delas é a implementação de dispositivos de segurança como *Intrusion Detection System* (IDS) e sistemas de firewall, que bloqueiam tráfegos potencialmente maliciosos. Contudo as técnicas de intrusão cresceram na mesma medida que as implementações de segurança, e os comandos e scripts de ataque se popularizaram, permitindo a qualquer indivíduo com algum conhecimento em tecnologia da informação encampar ataques contra redes ou servidores.

Nesse sentido, uma outra medida de segurança lógica que passou a ser adotada pelas organizações em complemento aos dispositivos de segurança foram os testes de penetração ou auditorias de segurança, que simulam inúmeras formas de ataque a uma rede com objetivo de encontrar vulnerabilidades que expõem a infraestrutura para, em seguida, mitiga-las.

1.1 TESTE DE INTRUSÃO

Conhecido também por teste de penetração, auditoria de segurança ou *pentest*, o teste de intrusão reúne um conjunto de ações adotadas por um profissional com vistas a identificar características como: vulnerabilidades conhecidas, má implementação de serviços, erros de procedimentos ou qualquer outro aspecto de torne um computador ou uma rede mais suscetível a ataques e consequente comprometimento.

Como trata-se de uma atividade extremamente técnica e abrangente, algumas instituições elaboraram guias que servem como poderoso auxílio aos *pentesters* (profissionais que executam esse tipo de teste) na execução de seu ofício, servindo como uma trilha balizadora que evita a não observância de alguma etapa.

Nos itens a seguir serão apresentados os principais guias, também denominados frameworks, utilizados pelos profissionais da área.

1.2 *FRAMEWORKS* DE TESTE DE SEGURANÇA

Um teste de penetração é uma atividade trabalhosa e complexa, pois envolve variados fatores que interagem para alcançar um produto simples de conceber, mas difícil de alcançar, a segurança. Conhecimento técnico, logística adequada, tempo, recursos financeiros, ferramentas adequadas, são alguns dos insumos presentes em um *pentest* que se apresenta como uma atividade com elevado grau de especificidade e tecnicismo.

Diante do desafio de padronizar esse procedimento, algumas instituições sugerem metodologias para nortear o trabalho dos profissionais de segurança, tornando o ofício menos espontâneo e mais sistêmico, colaborando para um trabalho mais eficiente e completo, evitando que algum aspecto relevante seja deixado de lado pela falta de perícia ou conhecimento do seu executante. Não há uma metodologia padronizada em escala global, mas algumas são aceitas internacionalmente pela sua maturidade, excelência e propositores, são elas: *Open Source Security Testing Methodology Manual* (OSSTMM 3), *Technical Guide to Information Security Testing and Assessment Penetration Testing Execution Standard* (PTES) e *Information Systems Security Assessment Framework* (ISSAF).

1.2.1 Open Source Security Testing Methodology Manual (OSSTMM 3)

O *Open Source Security Testing Methodology Manual* (OSSTMM 3) está em sua terceira versão, sendo mantido pelo *Institute for Security and Open Methodologies* (ISECOM), organização sem fins lucrativos. O projeto OSSTMM é aberto e por isso aceita colaboração de qualquer profissional com intenção e capacidade para colaborar com seu aperfeiçoamento. Essa metodologia é bastante completa e abarca avaliações de segurança sob cinco prismas: humano, físico, *wireless*, telecomunicações e redes de dados, indo além das necessidades impostas nesta pesquisa, que se concentra exclusivamente em redes de dados.

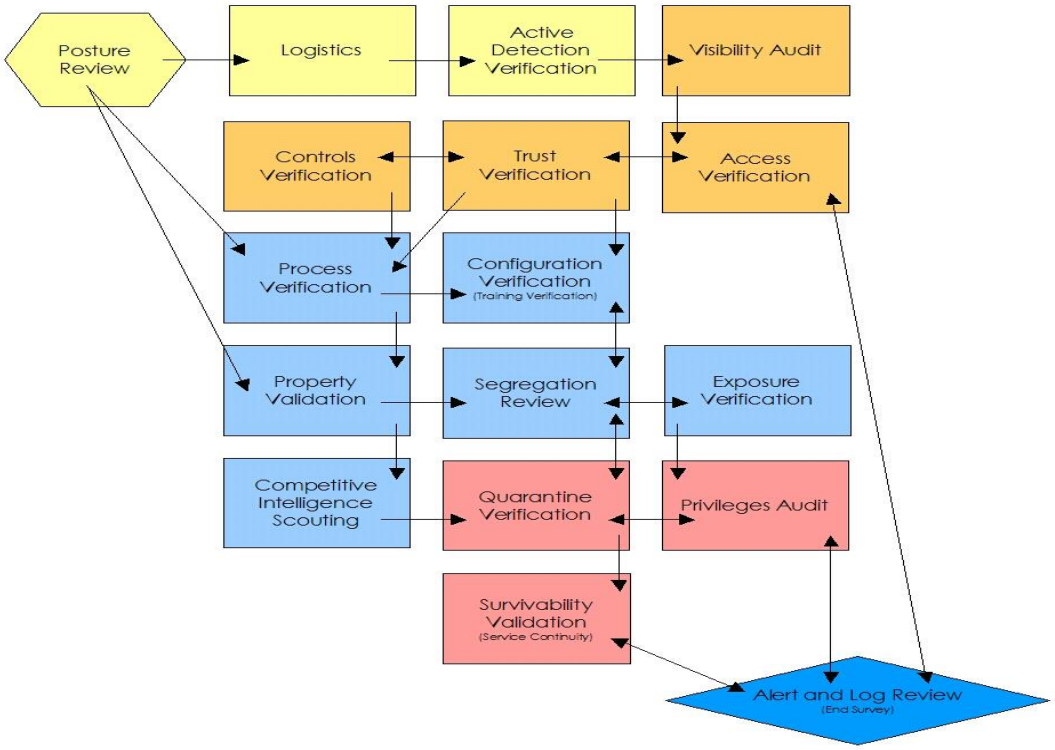
A metodologia OSSTMM conta com informações para o planeamento do projeto, quantificação dos resultados e as regras para realizar uma auditoria de segurança, com isso a sistemática pode ser integrada com leis e políticas já existentes para garantir uma auditoria de segurança completa na infraestrutura "alvo".

O grande objetivo do manual é garantir um processo amparado por um método científico, cujo resultado permita uma análise precisa e robusta de todos os aspectos de segurança de uma organização, podendo ser adotados em qualquer espécie de auditoria.

O fluxo de trabalho sugerido na metodologia em tela segue a seguinte hierarquia: 1.Canal; 2. Módulo; e 3. Tarefa.

Cada um dos cinco canais (humano, físico, *wireless*, telecomunicações e redes de dados) possuem os mesmos 17 módulos. Apesar dos módulos serem nomeados da mesma maneira independente do canal onde o teste esteja sendo executado, as tarefas são diferenciadas em cada um deles, recebendo entradas (*input*) e fornecendo saídas (*outputs*) específicas, que podem ser utilizadas como *input* de outro módulo. Dessa maneira, para uma atividade completa e eficaz, todos os módulos e tarefas devem ser realizados em sua integralidade para não haver prejuízo nas ações vindouras por falta de dados adequados.

Figura 01: Os 17 módulos da metodologia OSSTMM



Fonte: Herzog (2010, p. 103)

Na figura 01 os módulos estão representados por cores específicas. Os blocos amarelos tratam da *Induction Phase*, que conta com três módulos (*Posture Review, Logistics e Active Detecion Verification*). A fase seguinte, *Interaction Phase*, está destacada na cor laranja e comporta os seguintes módulos: *visibility audit, access verification, trust verification e control verification*. A quarta fase, nomeada como *Inquest Phase* abarca os módulos representados na cor azul e a última etapa, *Intervention phase* na cor vermelho claro.

Observando as direções das setas na figura é possível inferir que a metodologia em análise não segue uma sequência retilínea, mas aceita a adoção de "caminhos" alternativos e a repetição de fases já executadas, como pode ser observado na "caixa" *segregation review*, que permite ao pentester decidir entre seguir para qualquer um dos seguintes módulos: *configuration verification, exposure verification ou quarantine verification.*

Conforme descrito anteriormente, esse framework abarca cinco canais, indo além do propósito desta pesquisa. Dessa maneira, será apresentado um extrato apenas do canal "Redes de dados" no quadro a seguir, destacando os 17 módulos e suas respectivas tarefas, garantindo um entendimento holístico do manual em tela.

Quadro 01 – Detalhamento da metodologia OSSTMM

CANAL	MÓDULO	TAREFA
REDES DE DADOS	*Posture Review*	*Policy*
		Legislation and Regulations
		Culture
		Age
		Fragile Artifacts
	Logistics	*Framework*
		Network Quality
		Time
	Active Detection Verification	*Filtering*
		Active Detection
	Visibility Audit	*Network Surveying*
		Enumeration
		Identification
	Access Verification	*Network*
		Services
		Authentication
	Trust Verification	*Spoofing*
		Phishing
		Resource Abuse
	Controls Verification	*Non-repudiation*
		Confidentiality
		Privacy
		Integrity
	Process Verification	*Maintenance*
		Misinformation
		Due Diligence
		Indemnification
	Configuration Verification	*Configuration Controls*
		Common Configuration Errors
		Limitations Mapping
	Property Validation	*Sharing*
	Segregation Review	*Privacy Containment Mapping*
		Disclosure
		Limitations
		Discrimination
	Exposure Verification	*Exposure Enumeration*
	Competitive Intelligence Scouting	*Business Grinding*

		Profiling
		Business Environment
		Organizational Environment
	Quarantine Verification	*Containment Process Identification*
		Containment Levels
	Privileges Audit	*Identification*
		Authorization
		Escalation
	Survivability Validation	*Resilience*
		Continuity
		Safety
	Alert and Log Review	*Alarm*
		Storage and Retrieval

Fonte: O autor (2016)

Não serão definidas ou pormenorizadas cada uma das tarefas, devido à pouca relevância que tal ação traria para a pesquisa. De qualquer maneira, analisando a tabela acima, percebe-se pela vasta lista apresentada a granularidade da metodologia e o alto impacto esperado de sua correta execução.

1.2.2 Technical guide to information security testing and assessment

O *National Institute of Standards and Technology* (NIST) é responsável pelo desenvolvimento de normas e orientações nos Estados Unidos, incluindo os requisitos mínimos, para a prestação de segurança da informação adequada para todas as agências do governo estadunidense, sendo o *Technical Guide to Information Security Testing and Assessment* um dos produtos apresentados pelo NIST nessa área. Segundo consta no próprio guia, o seu escopo é fornecer orientações para as organizações sobre planejamento e realização de ensaios técnicos de informações de segurança e de avaliações, análise de resultados e desenvolvimento de estratégias de mitigação. Além disso, são oferecidas recomendações práticas para a concepção, implementação e manutenção de informação técnica relacionada com testes de segurança e processos e procedimentos de avaliação, que podem ser usadas para vários fins, tais como encontrar vulnerabilidades em um sistema ou rede e verificar o cumprimento de uma política ou outros requisitos.

O documento destina-se a equipes de segurança de TI e gestores de programas e sistemas, administradores de rede e outros profissionais que são responsáveis pelos aspectos técnicos de preparação, operação e segurança de sistemas e infra-estruturas de rede. Os

gerentes também podem usar as informações apresentadas no manual para facilitar os processos de tomada de decisões associadas com testes de segurança e avaliações

O manual abarca três grandes fases do pentes: *Review techniques (Documentation Review, Log Review, Ruleset Review, System Configuration Review, Network Sniffing, File Integrity Checking e Documentation Review); Target Identification and Analysis Techniques*

Network Discovery, Network Port and Service Identification, Vulnerability Scanning, Wireless Scanning); e Target Vulnerability Validation Techniques (Technique, Password Cracking, Social Engineering).

Finalizando o guia, são apresentados aspectos relativos ao planejamento, execução e atividades a serem desenvolvidas após a Avaliação de Segurança.

Em seu apêndice "A" o documento sugere duas distribuições Linux como ferramentas para um teste de segurança, o Back Track e o Knoppix STD, ambas são um compêndio de programas vocacionados a perícia computacional e testes de invasão.

1.2.3 Penetration Testing Execution Standard (PTES)

O padrão de execução de testes de penetração é constituido por sete fases principais, que cobrem todas as etapas relacionadas a um teste de penetração, inicando com o contato inicial com o "cliente" e raciocínio por trás de um pentest, passando pela coleta de informações e descoberta de vulnerabilidades a fim de obter uma melhor compreensão da organização avaliada.

A seguir estão listados os sete níveis propostos por este padrão: Interações pré-engajamento; Coleta de informações; Modelagem de ameaças; Análise de Vulnerabilidade; Exploração; Pós exploração; Publicar Exploração; Relatórios.

1.2.4 Information System Security Assessment Framework (ISSAF)

Desenvolvida pela *Open Information Systems Security Group* (OISSG), instituição sem fins lucrativos com sede em Londres, Inglaterra, essa metodologia de testes de penetração para avaliar redes, sistemas e aplicativos é constiuida por três fases e nove etapas.

As três fases gerais, que dividem-se nas demais ações são: Fase - I: Planejamento e Preparação; Fase - II: Avaliação; e Fase - III: Relatório, limpeza dos rastros e destruição dos artefatos. A primeira fase, planejamento e preparação, compreende as etapas para a troca de informação inicial, planejar e preparar para o teste. Nessa etapa, deve ser firmado um Acordo de Avaliação formal, que deve ser assinado entre as partes envolvidas para garantir o devido respaldo legal para as ações vindouras. Esse documento especifica ainda a equipe de trabalho,

as datas exatas, o tempo do teste e dá outras providências. Na etapa seguinte, de Avaliação, são realizados os testes propriamente ditos, seguindo a seguinte proposta: Coleta de Informações; Mapeamento de rede; Identificação de Vulnerabilidade; Penetração; Ganhar Acesso e escalação de privilégio; Enumerar usuários remotos e manter acesso. A ferramenta desenvolvida nesta pesquisa terá como foco apenas essa fase, devido a sua importância em um *pentest* e a possibilidade de automatização dos testes que a compõe. A terceira e última fase conciste em confecção do relatório, limpeza dos rastros e destruição dos artefatos, mas como dito anteriormente, essa fase não será coberta na ferramenta, exceto pela confecção do relatório, que será gerado seguindo as recomendações propostas pelo ISSAF.

Para desenvolver a ferramenta escopo dessa pesquisa será utilizada a metodologia sugerida pelo ISSAF, pois julgou-se que em detrimento dos demais "*frameworks*" apresentados ela possui foco mais dedicado a testes de penetração enquanto as outras possuem um viés mais holístico, abordando testes de segurança no sentido "*lato*" e por isso discutindo assuntos que extrapolam o objetivo proposto nesse trabalho, como segurança física. Além disso, a documentação confeccionada pelo ISSAF possui um refinamento bastante útil para o desenvolvimento de uma ferramenta automatizada, pois a cada fase da metodologia são apresentadas ferramentas que podem ser utilizadas para alcançar a finalidade desejada, além de uma descrição, análise e contramedidas indicadas para todas as etapas.

Para esta pesquisa será dado enfoque apenas na etapa de levantamento de informações de um *pentest*, com isso a metodologia ISSAF será adotada apenas nos tópicos referentes a levantamento de informações em detrimento das demais. Dessa maneira, serão implementados os seguintes tópicos previstos pela ISSAF: informating gathering, network mapping, vulnerability identification, todas referentes a etapa 2 (avaliação); e confecção do relatório, referente a etapa 3 (relatório, limpeza e destruição dos artefatos). Conforme citado anteriormente, a primeira etapa da metodologia se refere a preparação e planejamento, ações que não podem ser automatizadas e por isso fogem ao escopo deste trabalho, ficando de fora da ferramenta a ser desenvolvida.

1.3 LINGUAGEM DE PROGRAMAÇÃO E OUTROS *SOFTWARES*

Para desenvolver uma ferramenta de *pentest* é imperativo o uso de alguma linhagem de programação para automatizar as tarefas que seriam executadas manualmente por um pentester. Dentre as linguagens existentes foram escolhidas shell script e python para dar forma à ferramenta escopo da pesquisa.

A distribuição Linux utilizada para desenvolver o SAPIENS é a KALI, pois possui instalado por default uma série de ferramentas necessárias a realização de um pentest. Uma das maneiras de interagir com os diversos comandos *default* do KALI é através de scripts em shell, por isso essa linguagem foi eleita para ser utilizada. Por outro lado, a linhagem python será empregada devido as inúmeras bibliotecas que possui, permitindo funcionalidades como desenvolvimento web e integração com outras linguagens como o próprio shell script.

1.3.1 Kali Linux

O Kali é uma distribuição Linux Open Source fundada e mantida pela Offensive Security, empresa que presta serviços de capacitação em segurança da informação e testes de penetração[3]. Essa versão do SO Linux tem como objetivo ser um framework para atividades relacionadas à ataque e segurança da informação, contando com inúmeras ferramentas de coleta de informações, escaneamento, enumeração, exploração, manutenção de acesso e limpeza de rastros.

1.3.2 Shell Script

Shell é a camada mais externa no sistema operacional UNIX, responsável pela interação entre o usuário e kernel. Em outras palavras shell é o intermediário que traduz o que é digitado através do teclado e transmite ao sistema operacional, contando para isso com uma linguagem interpretada de alto nível, que permite a construção de loop, tomada de decisão e armazenamento de valores em variáveis.[4]

Neves (2001) lista as seguintes tarefas do shell: exame da linha de comando identificando a existência de comandos, opções, parâmetros, redirecionamentos e variáveis; atribuição de variável; Resolução de redirecionamentos; substituição de variáveis; substituição de metacaracteres e envio da linha de comando para o *kernel*.

---

[3] Disponível em: https://www.kali.org/about-us/ Acesso em: 21 mar 2016

[4] NEVES, Julio Cezar. **Linux**: programação shell. 2. ed. Rio de Janeiro: Brasport, 2001.

1.3.3 Python

A linguagem Python possui uma sintaxe clara e objetiva, que facilita o entendimento do código até pelos programadores menos experientes. A linguagem conta com inúmeras estruturas de alto nível (dicionários, listas, data / hora, complexos e outras) além de módulos e *frameworks* de terceiros que podem ser inseridos. Possui ainda "ferramentas" presentes na maioria das linguagens, como: geradores, introspecção, persistência, metaclasses e unidades de teste.[5]

Borges (2015, p. 14) apresenta uma visão geral sobre Python:

> Multiparadigma, a linguagem suporta programação modular e funcional, além da orientação a objetos. Mesmo os tipos básicos no Python são objetos. A linguagem é interpretada através de *bytecode* pela máquina virtual Python, tornando o código portável. Com isso é possível compilar aplicações em uma plataforma e rodar em outros sistemas ou executar direto do código fonte. (...). Além de ser utilizado como linguagem principal no desenvolvimento de sistemas, o Python também é muito utilizado como linguagem *script* em vários softwares, permitindo automatizar tarefas e adicionar novas funcionalidades, entre eles: BrOffice.org, PostgreSQL, Blender, GIMP e Inkscape. É possível integrar o Python a outras linguagens, como a Linguagem C e Fortran.

Dentre as características listas por Borges (2015) destaca-se, para essa pesquisa, a capacidade da linguagem se integrar com outras, como Fortran, C e principalmente Shell Script.

Com base no exposto fica evidente as facilidades introduzidas pela linguagem e a multifuncionalidade, característica indispensável para o escopo proposto nessa pesquisa.

1.4 INTEGRAÇÃO COM OUTROS *SOFTWARES*

As ações de um teste de penetração contam com uma série de ferramentas amplamente utilizadas que atuam como facilitadores indispensáveis aos profissionais de segurança da informação. Algumas cuidam apenas de parte do processo de auditoria, como o Selemuin, que é utilizado predominantemente para avaliação em aplicações web. Outras ferramentas surgem com uma solução mais holística, avaliando a rede ou máquinas alvo de maneira mais completa, nesse grupo recebe grande destaque o software Nessus, que gera um relatório de auditoria bastante completo. Tal ferramenta assemelha-se com o SAPIENS, guardando, entretanto, características que reforçam a necessidade do desenvolvimento de uma nova ferramenta automatizada.

---

[5] BORGERS, Luiz E. **Python para Desenvolvedores.** 1. ed. São Paulo: Novatec, 2015

Diante disso e levando em conta o resultado do questionário de percepção de opinião distribuído a profissionais da área de segurança, que será apresentado e discutido anteriormente, surgiu a necessidade de integrar o SAPIENS com algumas ferramentas para potencializar seus resultados. Dessa maneira, o software de automatização de pentest, escopo dessa pesquisa, foi associado, principalmente, às seguintes ferramentas: nmap, nessus e selenium.

1.4.1 Nmap

A principal vocação do NMAP é realizar scaneamento em hosts afim de descobrir o estado das portas lógicas e os serviços em execução. Contudo, esse software está apto a contribuir em muitos outros aspectos, conforme citado em seu site oficial a ferramenta é útil para auditoria, gerência, inventário e segurança de redes. Com NMAP é possível identificar ainda hosts ativos em uma rede e quais serviços determinado host oferece, além do sistema operacional que está instalado. Outra possibilidade é descobrir o tipo de firewall está em uso na rede.[6]

Buscando uma maneira de utilizar o NMAP de forma mais eficiente, foi empregado no SAPIENS uma funcionalidade desse software conhecida como "Nmap Scripting Engine" (NSE), que agrupa uma série de comandos em categorias distintas, permitindo ao usuário executar inúmeras interações com o host alvo executando poucas linhas de comando. As categorias disponíveis para uso são: *auth, broadcast, default. discovery, dos, exploit, external, fuzzer, intrusive, malware, safe, version, and vuln*. Para a integração com o SAPIENS, julgou-se desnecessário o uso de todas as categorias de scaneamento, pois algumas são redundantes o que acarretaria em desnecessário consumo de recurso computacional, além de aumentar enormemente o tempo de execução. Com isso, foram implementados na SAPIENS os scripts *default, discovery e exploit*.

As três categorias eleitas possuem as seguintes características: *exploit* – conjunto de comandos que tenta explorar algumas vulnerabilidades; *discovery* – tenta descobrir ativamente mais sobre a rede consultando registros públicos; *default* – inclui scan que devem terminar rapidamente, produzir informações úteis e enxutas e ser pouco intrusivo.[7]

[6] Disponível em: https://nmap.org/. Acesso em 11 fev 2016.
[7] Disponível em: https://nmap.org/book/nse-usage.html#nse-categories. Acesso em 11 fev 2016.

1.4.2 Nessus

O Nessus é um *scanner* de vulnerabilidade mantido e comercializado pela empresa Tenable Network Security. O *scan* utiliza a topologia cliente-servidor, sendo que o resultado é apresentado na tela do cliente e as ações propriamente ditas são executadas pelo servidor.

A ferramenta utiliza uma interface web e apresenta ao usuário um relatório detalhado com as vulnerabilidades conhecidas de uma máquina específica ou uma rede alvo. Uma funcionalidade bastante útil são os plug-ins que permitem a integração com outros softwares como o Metasploit.

O produto é comercializado em diferentes versões: Nessus Cloud, Manger, Professional e Home, sendo esta última a única gratuita, mas com limitação de apenas 16 endereços IP e limitada ao uso doméstico para fins não comerciais.[8]

1.4.3 Selenium

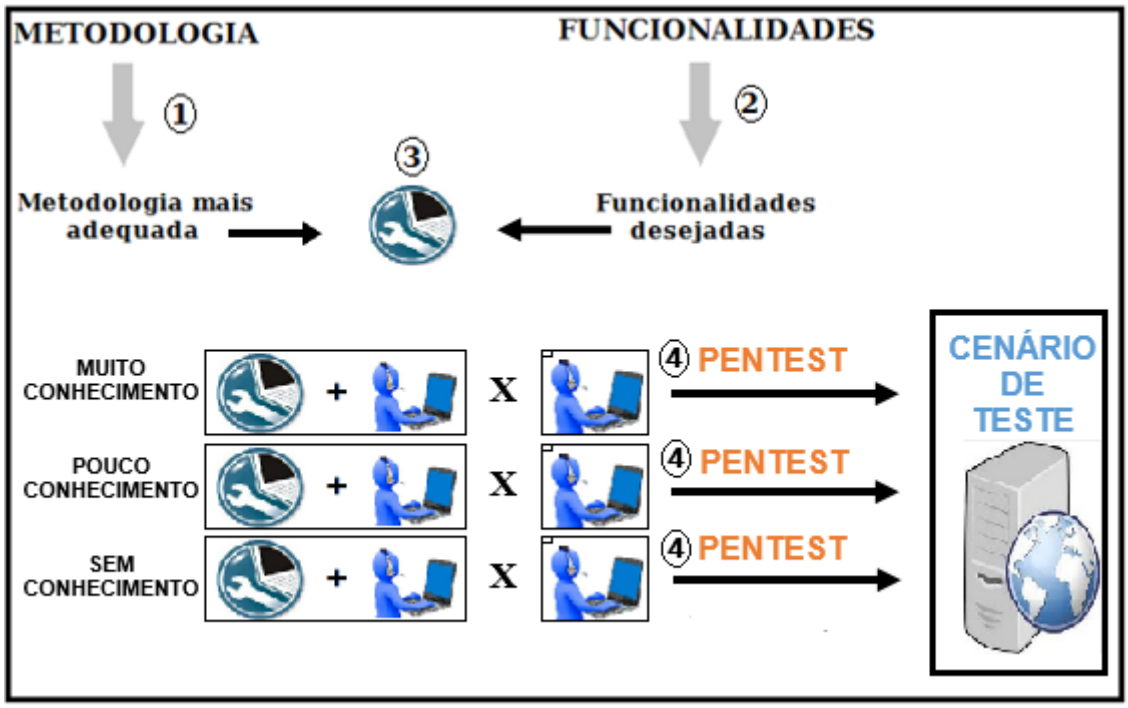
O Selenium é um conjunto de ferramentas dedicado a automação de teste, com foco principal em aplicações web. O domínio completo dessa suíte de ferramentas permite ao profissional de segurança grande flexibilidade na execução de sua auditoria através da comparação do comportamento esperado de uma aplicação com seu funcionamento real em determinada situação. Para integrar com o SAPIENS foi utilizada o Selenium IDE (Integrated Development Environment), um plugin do browser Firefox que fornece uma interface amigável para auxiliar a construção de scripts. Esse plugin possui uma funcionalidade que grava as interações do usuário com o browser e as exporta como script em diversas linguagens de programação, como: java, csharp, Ruby, php, perl, javascript e python, sendo esta última a adotada nessa pesquisa.[9]

[8] Disponível em: https://www.tenable.com/products/nessus-vulnerability-scanner. Acesso em 21 mar 2016.
[9] Disponível em: http://www.seleniumhq.org/projects/ide/. Acesso em: 11 fev 2016.

2 METODOLOGIA

Para alcançar o escopo apresentado e testar as hipóteses delimitadas a presente pesquisa seguiu dois caminhos convergentes e complementares, integrando teoria e prática. Com o intuito de garantir um sólido arcabouço metodológico à ferramenta proposta, dentre algumas metodologias de *pentest* propostas por instituições conceituadas na área de segurança, foi selecionado a metodologia do ISSAF como modelo mais adequado a ser seguido, servindo de base para a construção do SAPIENS. As etapas desta metodologia que serão implementadas na ferramenta serão: informating gathering, network mapping, vulnerability identification, todas referentes a etapa 2 (avaliação); e confecção do relatório, referente a etapa 3 (relatório, limpeza e destruição dos artefatos). Tomando por base a figura 2, que ilustra graficamente os passos que forma executados para dar forma a esse trabalho, a etapa supracitada atende ao número "(1)".

Figura 2: Metodologia de pesquisa adotada neste trabalho



Fonte: O autor (2016)

Em seguida, foram distribuídos questionários a profissionais da área de segurança para analisar a percepção deles e identificar funcionalidades úteis ao ofício do auditor de redes de computadores na realização de um teste de penetração e que poderiam ser incorporadas ao SAPIENS, essa etapa está caracterizada na figura acima pelo número "(2)". A apreciação dos resultados desses questionários será apresentada no capítulo a seguir.

A etapa identificada com o número (3) consistiu no desenvolvimento efetivo do software para automatizar as ações de *pentest*. Foi nesta etapa que a ferramenta SAPIENS tomou forma e foi efetivamente desenvolvida, levando-se em conta a metodologia definida no passo "(1)" e as opiniões dos profissionais de segurança coletadas na fase "(2)".

É possível observar ainda na figura acima a fase representada pelo número "(4)", quando o *software* desenvolvido foi posto à prova através de uma avaliação de desempenho em um ambiente controlado no intuito de mensurar as vantagens introduzidas pela solução. Para tanto, foram selecionados três pares de voluntários agrupados de acordo com seu grau de conhecimento e experiência na área de tecnologia e/ou segurança da informação. Todas as três duplas realizaram uma auditoria de segurança em um cenário de teste (igual para todos), sendo que um dos voluntários de cada dupla utilizou o SAPIENS e o outro não. O intuito foi mensurar as vantagens e prejuízos acarretados com o uso da ferramenta entre profissionais de diferentes níveis de formação e experiência durante um *pentest*.

Para obter a heterogeneidade desejada entre os voluntários, foram selecionados profissionais com diferentes perfis e agrupados da seguinte maneira: a primeira dupla, rotulada na figura acima como "MUITO CONHECIMENTO", era composta por profissionais com graduação ou pós na área de tecnologia da informação e mais de 2 anos de experiência no ramo de segurança; a dupla seguinte (POUCO CONHECIMENTO) era formada por indivíduos com alguma formação na área de segurança da informação, porém sem nenhuma experiência prática; e, por fim, o último par era constituído por profissionais de outras áreas, sem qualquer ligação com tecnologia da informação e/ou segurança e por isso foram rotulados como "SEM CONHECIMENTO".

Ao final do experimento os seis relatórios de pentest gerados por cada um dos voluntários foi comparado, levando-se em conta as seguintes variáveis: utilizou a ferramenta SAPIENS ou não; grau de conhecimento da área de tecnologia e/ou segurança da informação; quantidade de servidores encontrados no cenário de teste; quantidade de nomes de funcionários encontrados; quantidade de e-mail encontrados; quantidade de máquinas ativas encontradas; quantidade de vulnerabilidades encontradas. O modelo de relatório preenchido pelas duplas encontra-se no apêndice B desta pesquisa.

Para os três profissionais que realizaram os teste utilizando a ferramenta SAPIENS (um de cada dupla) foram coletadas ainda as respectivas opiniões sobre o desempenho e resultado do software, o objetivo foi apurar se o mesmo atendeu os anseios da população que respondeu o questionário na etapa "(2)".

Com o cruzamento dos dados coletados buscou-se testar a efetividade e eficácia da solução proposta entre profissionais com diferentes níveis de conhecimento, alcançando o escopo proposto para a pesquisa que é: desenvolver uma ferramenta para tornar as tarefas de um teste de penetração mais céleres e metodológicas, garantindo uma eficiência mais refinada através da minimização do fator humano nas ações de descoberta e exploração de vulnerabilidades em sistemas informáticos.

2.1 PESQUISA DE CAMPO SOBRE A PERCEPÇÃO DAS FUNCIONALIDADES NECESSÁRIAS EM UMA FERRAMENTA AUTOMATIZADA DE TESTES DE PENETRAÇÃO

A pesquisa de campo foi conduzida para levantar informações complementares sobre as funcionalidades desejadas na ferramenta SAPIENS, para tanto foi selecionada uma amostra e em seguida a população para receberem os questionários piloto e definitivo, respectivamente.

O objetivo da distribuição de um questionário piloto foi corrigir eventuais falhas formais e de conteúdo, além de possibilitar a adoção de melhorias nas questões apresentadas, fruto de opiniões de terceiros. Essa ferramenta foi distribuída em novembro de 2015 para militares da Seção de Guerra Cibernética do Centro de Instrução de Guerra Eletrônica (CIGE), unidade do Exército Brasileiro responsável por conduzir as ações de capacitação dos militares das Forças Armadas na área de cibernética. Após as correções e sugestões apontadas por esse grupo teste, foi produzido o questionário definitivo.

Enviado a militares das Forças Armadas, o instrumento de pesquisa definitivo, já com as alterações, buscou obter insumos inexistentes na bibliografia estudada e que só poderiam ser alcançados através da análise da opinião de profissionais que dedicam-se a tarefa de auditor redes de computadores.

Os questionários foram distribuídos em dezembro de 2015 e no total, responderam-no 23 militares.

2.1.1 População selecionada

A população selecionada para responder o questionário em tela pertence a um restrito grupo de militares das Forças Armadas concludentes dos Cursos de Guerra Cibernética para Oficias e Sargentos, ministrado pelo CIGE, cujo escopo principal é habilitar oficiais e

sargentos para a ocupação de cargos e o desempenho de funções de segurança, defesa e guerra cibernética. [10]

2.2 DESENVOLVIMENTO DO SOFTWARE SAPIENS

Para desenvolver o SAPIENS foi utilizada uma máquina virtual com 40 Gigabits de memória interna e 2 Gigabits de memória RAM (*Random Access Memory*). O virtualizador utilizado foi o software VMware Workstation, versão 10.0.6 build-2700073.

A máquina virtual possui o seguinte sistema operacional e versão do Kernel: Linux kali000 4.0.0-kali1-686-pae #1 SMP Debian 4.0.4-1+kali2 (2015-06-03) i686 GNU/Linux. Para viabilizar as ações necessárias ao pleno funcionamento da ferramenta desenvolvida, foram instalados os seguintes pacotes que não são *default* do Kali: ipcalc, sipcalc, httrack, nessus e vfeed. Além disso, foi instalado o plugin "Selenium IDE" no browser Iceweasel e gerada uma credencial de acesso para a ferramenta Nessus.

2.3 CENÁRIO DE TESTES

Para as três duplas testarem a ferramenta SAPIENS foi disponibilizado o seguinte cenário de teste: 38.100.193.76 (www.megacorpone.com), 38.100.193.83 (admin.megacorpone.com), 38.100.193.69 (beta.megacorpone.com), 38.100.193.82 (fs1.megacorpone.com), 38.100.193.90 (ns3.megacorpone.com), 38.100.193.84 (mail.megacorpone.com), 38.100.193.80 (ns2.megacorpone.com), 38.100.193.81 (intranet.megacorpone.com), 38.100.193.77 (vpn.megacorpone.com), 38.100.193.66 (syslog.megacorpone.com), 38.100.193.69 (beta.megacorpone.com), 38.100.193.70 (ns1.megacorpone.com), 38.100.193.72 (admin.megacorpone.com), 38.100.193.73 (mail2.megacorpone.com), 38.100.193.80 (ns2.megacorpone.com), 38.100.193.84 (mail.megacorpone.com), 38.100.193.85 (snmp.megacorpone.com), 38.100.193.89 (siem.megacorpone.com), 38.100.193.90 (ns3.megacorpone.com), 38.100.193.91 (router.megacorpone.com).

Toda esta infraestrutura pertence à *Ofensive Security*, empresa responsável por cursos na área de segurança da informação, como: *Offensive Security Certified Professional* (OSCP) e *Pentest with Kali* (PWK).

---

[10] BRASIL. Estado-Maior do Exército Brasileiro. **Portaria Nº 185:** Cria o Curso de Guerra Cibernética para Sargentos e estabelece suas condições de funcionamento. Brasília, 2012.

Como parte destes cursos a *Ofensive Security* criou uma empresa fictícia denominada MEGACORPONE, que possui servidores e serviços ativos na internet preparados para serem auditados pelos alunos dos seus diversos cursos. Como se trata de um ambiente preparado para ser testado e com vulnerabilidades mapeadas, o ambiente constituiu-se como um excelente cenário de teste para a presente pesquisa.

Além da infraestrutura citada, foi utilizada também uma máquina com sistema operacional Linux, que serviu como alvo específico para as ações de *scanning*, enumeração e levantamento de vulnerabilidades. Essa máquina possui as seguintes características: linux metasploitable 2.6.20-16-server # 1 SMP i 686 GNU/Linux com os seguintes; os seguintes serviços ativos: samba 3.8, apache Jserv, postgreSQL 8.3 e Linux telnetd.

3 RESULTADO E ANÁLISES

Este capítulo apresenta os resultados obtidos com os questionários de percepção de opinião distribuídos aos profissionais de segurança da informação e dos testes de auditoria realizados com seis voluntários, onde a ferramenta SAPIENS foi efetivamente testada.

3.1 QUESTIONÁRIO DE PERCEPÇÃO DE OPINIÃO

A partir deste questionário, a população de profissionais forneceu os dados que permitiram compreender a opinião destes acerca das características desejáveis em uma ferramenta de teste de penetração. A identificação dos que preencheram os questionários foi facultada com o intuito de garantir maior liberdade nas suas respostas.

3.1.1 Experiência Profissional

Neste item o profissional deveria responder quantos anos de experiência possui na área de segurança.

Tabela 01 – Experiência profissional

Menos de 1 ano	4	17.4%
Mais de 1 e menos de 3 anos	8	34.8%
Mais de 3 e menos de 5 anos	8	34.8%
Mais de 5 anos	3	13%
TOTAL	23	

Fonte – O autor, 2016

Com base na tabela 01 é possível perceber que 47.8% dos participantes possui mais de 3 anos de experiência na área de estudo dessa pesquisa, referendando a relevância da opinião da população analisada.

3.1.2 Influência na realização de testes de penetração

Com o intuito de identificar as causas que limitam a execução dos testes de penetração nas unidades da Forças Armadas foi perguntado quais fatores inibem a realização de testes de penetração nas empresas e instituições.

Não foi delimitado um número máximo de opções selecionadas, permitindo ao participante escolher no mínimo 1 e no máximo 8 opções

Tabela 02 – Fatores que inibem a realização dos testes de penetração

Os testes de penetração são atividades popularizadas e largamente empregadas em empresas e organizações	0	0%
Falta de profissionais capacitados para o ofício	13	56.5%
Custo financeiro da atividade	6	26.1%
Falta de preocupação com segurança	14	60.9%
Ineficiência desse tipo de teste	0	0%
Desconhecimento desse tipo de teste	12	52.2%
Outras medidas de segurança substituem o *pentest*	0	0%
Receio de expor negativamente o nome da instituição	17	73.9%
Outros	0	0%

Fonte – O autor, 2016

É possível perceber na tabela 02 que, segundo a reposta do grupo analisado, ninguém julga o *pentest* como algo ineficiente ou substituível por outras medidas, reforçando a relevância do tema dessa pesquisa. O percentual de 73.9% acredita que essa atividade é limitada devido ao receio de expor negativamente o nome da instituição. Outras respostas expressivas foram: falta de preocupação com segurança, falta de profissionais capacitados para o ofício e desconhecimento desse tipo de teste, todas as respostas com mais de 50% de incidência.

3.1.3 Características de um software de pentest automatizado

Foi apresentada a seguinte pergunta com o objetivo de levantar requisitos para o desenvolvimento de um software automatizada de pentest: em uma ferramenta de automação de testes de penetração, escolha as 4 (QUATRO) características que o senhor julga mais relevante nesse tipo de software.

Como todas as opções de reposta traziam características potencialmente úteis e devido a necessidade de estabelecer prioridades, julgou-se necessário limitar a 4 a quantidade de respostas de cada participante, dessa maneira este ficaria obrigado a priorizar as características mais relevantes em detrimento das demais.

Tabela 03 – Características desejáveis em uma ferramenta de pentest

Interface Gráfica amigável	7	30.4%
Relatório detalhado e com sugestões de correção	19	82.6%
Possibilidade de trabalho colaborativo entre dois ou mais profissionais simultaneamente	10	43.5%
Possibilidade de executar a ferramenta remotamente, sem a necessidade de estar na rede alvo	6	26.1%
Integração com outras ferramentas, como Nessus, Metasploit e GFI Languard.	10	43.5%
Segurança dos dados obtidos e do canal por onde essas informações trafegam	11	47.8%
Possibilidade de executar ações com o mínimo de interação do usuário, possibilitando a realização do pentest por empresas e profissionais sem tantos recursos e/ou conhecimento técnico	2	8.7%
Garantir anonimato nas ações desenvolvidas	5	21.7%
Escalabilidade, ou seja, permitir a integração com novos scripts ou funcionalidades desenvolvidas pelos usuários	14	60.9%
Outros	1	4.3%

Fonte – O autor, 2016

Conforme apresentado na tabela 03, as quatro funcionalidades mais necessárias segundo os profissionais ouvidos são: relatório detalhado e com sugestões de correção (82.6%); escalabilidade, ou seja, permitir a integração com novos scripts ou funcionalidades desenvolvidas pelos usuários (60.9%); Segurança dos dados obtidos e do canal por onde essas informações trafegam (47.8%); e Integração com outras ferramentas, como Nessus, Metasploit e GFI Languard (43.5%).

Com base nesse resultado tentou-se implementar essas quatro características no SAPIENS, deixando de lado as demais devido a premissa de tempo que limita esta pesquisa.

3.1.4 Periodicidade das Auditorias de Segurança

Com o intuito de estimar a periodicidade dos testes de penetração realizados nas Forças Armadas e identificar como são executadas, foi perguntado se os órgãos/empresas que os profissionais trabalhas realizam pentest ao menos uma vez por ano.

Tabela 04 – Periodicidade das auditorias de segurança

Sim, contrata profissionais externos para realizar esse tipo de atividade.	0	0%
Sim, é conduzida exclusivamente por funcionários internos	8	36.4%
Não, durante o período que estou na empresa nunca foi realizada nenhuma avaliação dessa natureza	10	45.5%
Não, são realizadas algumas auditorias, mas com baixa frequência (menos de uma vez por ano)	3	13.6%
Desconheço	1	4.5%
TOTAL	22	

Fonte – O autor, 2016

Diante do resultado é possível perceber que a maioria das organizações militares (59.1%) não realizam *pentest* de maneira regular (ao menos uma vez ao ano). Esse indicador apresenta uma questão preocupante, pois uma auditoria de segurança na rede é uma ação desejável para incrementar segurança, revelando potenciais vulnerabilidades desconhecidas pelos administradores.

Com base na constatação acima, resta saber os motivos que acarretam esse indicador. A pergunta seguinte apresentada no questionário tenta lançar uma luz nessa questão, elucubrando se a existência de uma ferramenta automatizada poderia reduzir o percentual apresentado.

3.1.5 Expectativa Sobre A Ferramenta

Na última questão do instrumento de coleta de opinião, foi perguntado se a existência de uma ferramenta automatizada para realizar testes de penetração pela empresa, aumentaria a frequência dos testes de penetração.

Tabela 05 – Impacto do uso de uma ferramenta automatizada

Sim	19	82.6%
Não	2	8.7%
Indiferente	1	4.3%
Minha empresa/instituição já possui uma ferramenta dessa natureza	0	0%
Outros	1	4.3%
TOTAL	23	

Fonte – O autor, 2016

As respostas revelam que a imensa maioria dos profissionais (82.6%) julgou que a posse de uma ferramenta automatizada traria um impacto positivo no que se refere a segurança da rede de computadores das organizações, incrementando a quantidade de *pentests* realizados ao longo de um ano. Esse índice reforça a necessidade por parte das instituições, traduzida pela opinião dos seus profissionais da área de segurança, de possuir uma solução que execute testes de penetração, pois a ações manuais são pouco eficientes e escaláveis.

Dessa maneira, o desenvolvimento do SAPIENS mostra-se ainda mais relevante, pois sua proposta é justamente suprir essa demanda, servindo como um auxílio na busca por uma rede de dados mais segura. O Código fonte da ferramenta está disponibilizado e comentado no apêndice C desta pesquisa.

3.2 TESTES DE AUDITORIA

A concepção de um ambiente de teste deu-se devido a necessidade de testar a efetividade e eficácia do SAPIENS em uma auditoria de segurança. Para mensurar as eventuais vantagens e prejuízos da ferramenta entre indivíduos com diferentes níveis de conhecimento em segurança da informação, os voluntários foram divididos em três pares agrupados de acordo com a formação acadêmica e experiência na área.

O par rotulado como “MUITO CONHECIMENTO” possui voluntários com, no mínimo, uma pós-graduação relacionada à segurança da informação e mais de três anos de experiência em testes de penetração. A dupla com “POUCO CONHECIMENTO” também possui pós-graduação na área, contudo sem experiência em *pentest*. Por fim, a última dupla não possui formação acadêmica, nem experiência em atividades de auditoria de segurança.

3.2.1 Quantidade de servidores encontrados

Neste item os voluntários deveriam apresentar quantos servidores relacionados ao domínio MEGACORPONE.COM encontraram, listando os respectivos endereços IP.

Tabela 06 – Servidores Encontrados

	Quantidade de servidores encontrados	Quantidade de servidores existentes	Taxa de acerto
Voluntário com **MUITO CONHECIMENTO** Utilizando a ferramenta SAPIENS	26	26	100%
Voluntário com **MUITO CONHECIMENTO** Sem o uso da ferramenta SAPIENS	16		61,53%
Voluntário com **POUCO CONHECIMENTO** Utilizando a ferramenta SAPIENS	26		100%
Voluntário com **POUCO CONHECIMENTO** Sem o uso da ferramenta SAPIENS	03		11,53%
Voluntário **SEM CONHECIMENTO** Utilizando a ferramenta SAPIENS	20		76,92%
Voluntário **SEM CONHECIMENTO** Sem o uso da ferramenta SAPIENS	02		7,69%

Fonte – O autor, 2016

Os voluntários que utilizaram o SAPIENS alcançaram resultados bastante superiores aos demais. Aqueles com POUCO CONHECIMENTO e SEM CONHECIMENTO obtiveram um percentual ainda mais significativo quando comparados aos seus respectivos pares que não lançaram mão da ferramenta.

A taxa média de acerto dos três voluntários que utilizaram a ferramenta foi de 92,3% e daqueles que não utilizaram foi de 26,91%.

3.2.2 Quantidade de Nomes de Funcionários Encontrados

Foi perguntado quantos nomes de funcionário da empresa alvo os voluntários encontraram. Esse dado é útil para eventuais ataques de engenharia social.

Tabela 07 – Nomes de Funcionários Encontrados

	Quantidade de nomes de funcionários encontrados	Quantidade de nomes de funcionários encontrados	Taxa de acerto
Voluntário com **MUITO CONHECIMENTO** Utilizando a ferramenta SAPIENS	01	04	25%
Voluntário com **MUITO CONHECIMENTO** Sem o uso da ferramenta SAPIENS	03		75%
Voluntário com **POUCO CONHECIMENTO** Utilizando a ferramenta SAPIENS	01		25%
Voluntário com **POUCO CONHECIMENTO** Sem o uso da ferramenta SAPIENS	03		75%
Voluntário **SEM CONHECIMENTO** Utilizando a ferramenta SAPIENS	0		0%
Voluntário **SEM CONHECIMENTO** Sem o uso da ferramenta SAPIENS	03		75%

Fonte – O autor, 2016

Comparando o desempenho entre as duplas, é possível observar na tabela acima uma expressiva diferença entre os usuários do SAPIENS e os demais, com clara desvantagem para os primeiros. Uma hipótese sugerida para explicar os dados é a fonte desse tipo de informação. Os nomes de funcionários foram obtidos através da leitura das informações contidas na página web da empresa alvo, o que demanda uma capacidade analítica mais refinada, característica que falta para o SAPIENS. Para dirimir essa ineficiência poderiam ser criados filtros com os nomes mais comuns em um determinado país, de forma que esses padrões fossem buscados na página web da empresa alvo. Contudo ainda assim não seria possível afirmar com segurança que os nomes encontrados são de terceiros ou de funcionários. Sendo assim, fica evidente a limitação da ferramenta neste ponto.

A taxa média de acerto dos três voluntários que utilizaram a ferramenta foi de 16,66 % e daqueles que não utilizaram foi de 75%.

3.2.3 Quantidade de E-mails Relacionados ao Domínio

Nesta questão o voluntário deveria procurar por endereços de e-mail relacionados ao domínio alvo. Assim como o item anterior, este tipo de dado é útil para ataques de engenharia social.

Tabela 08 – Endereços de e-mail Encontrados

	Quantidade de e-mails encontrados	Quantidade de e-mails encontrados	Taxa de acerto
Voluntário com **MUITO CONHECIMENTO** Utilizando a ferramenta SAPIENS	11	16	68,75%
Voluntário com **MUITO CONHECIMENTO** Sem o uso da ferramenta SAPIENS	07		43,75%
Voluntário com **POUCO CONHECIMENTO** Utilizando a ferramenta SAPIENS	11		68,75%
Voluntário com **POUCO CONHECIMENTO** Sem o uso da ferramenta SAPIENS	03		18,75%
Voluntário **SEM CONHECIMENTO** Utilizando a ferramenta SAPIENS	11		68,75%
Voluntário **SEM CONHECIMENTO** Sem o uso da ferramenta SAPIENS	06		37,5%

Fonte – O autor, 2016

É possível observar na tabela que nenhum dos voluntários alcançou uma taxa de acerto superior a 70%, contudo nota-se certa vantagem para os que utilizaram o SAPIENS, com um hiato mais acentuado entre as duplas com menos experiência e conhecimento.

Assim como ocorreu com o item anterior, a ferramenta falhou na busca por informações na página web da empresa. Todavia neste caso é possível alterar seu código e inserir um filtro eficiente na identificação de e-mails válidos, pois existe um padrão relativamente simples de ser encontrado, bastando que o filtro "case" com os caracteres XXX@DOMINIOALVO.COM. Ou seja, a efetividade da ferramenta neste item pode ser incrementada.

A taxa média de acerto dos três voluntários que utilizaram a ferramenta foi de 68,75 % e daqueles que não utilizaram foi de 33,3%.

3.2.4 Máquinas Ativas Encontrados

Em complemento ao item 01 deste questionário, nesta questão foi solicitado que o "pentester" identificasse as máquinas ativas na rede alvo, garantindo um rol de possíveis alvos para ataques cibernéticos.

Tabela 09 – Máquinas ativas

	Quantidade de máquinas ativas encontradas	Quantidade de máquinas ativas existentes	Taxa de acerto
Voluntário com **MUITO CONHECIMENTO** Utilizando a ferramenta SAPIENS	17	37	45,94%
Voluntário com **MUITO CONHECIMENTO** Sem o uso da ferramenta SAPIENS	37		100%
Voluntário com **POUCO CONHECIMENTO** Utilizando a ferramenta SAPIENS	17		45,94%
Voluntário com **POUCO CONHECIMENTO** Sem o uso da ferramenta SAPIENS	01		2,7%
Voluntário **SEM CONHECIMENTO** Utilizando a ferramenta SAPIENS	05		13,51%
Voluntário **SEM CONHECIMENTO** Sem o uso da ferramenta SAPIENS	00		0%

Fonte – O autor, 2016

Analisando os dados da tabela salta aos olhos o resultado alcançado pelo voluntário com MUITO CONHECIMENTO que não usou o software SAPIENS. Todavia o teste para validar os endereços IP ativos listados pelo voluntário revelou que a grande maioria não apresentava sinais de atividade, ou seja, não respondia às consultas com o protocolo ICMP ou TCP. Em contrapartida, os 17 endereços listados pelos usuários do SAPIENS revelaram atividade por ocasião da contraprova.

Partindo do princípio que as 37 máquinas foram encontradas por um profissional com conhecimento e experiência em testes de penetração incapaz de cometer um erro dessa natureza, uma hipótese aceitável é que, por ocasião do *pentest* as máquinas estavam ativas e durante a contraprova não estavam mais.

De qualquer maneira, o resultado deste item torna-se pouco robusto, pois não é possível adotar um parâmetro de comparação sólida, haja vista que alguma mudança pode ter ocorrido no cenário de teste.

Outro ponto que desperta interesse é o resultado apresentado pelo voluntário SEM CONHECIMENTO com uso do SAPIENS, que listou apenas 05 máquinas ativas contra 17 dos demais usuários da ferramenta, um percentual quase 35% inferior. A questão que enseja análise decorre do fato dos relatórios obtidos pelos três usuários da ferramenta terem o relatório exatamente igual nesse ponto, apresentando as mesmas 17 máquinas encontradas.

Resta saber o motivo que levou o voluntário a apresentar em sua resposta apenas 05 dos 17 IP encontrados.

Tal fato sugere que para extrair todas as informações transcritas no relatório do SAPIENS, o profissional deve ter algum conhecimento em auditoria de segurança, caso contrário dados relevantes podem ser desprezados pela simples falta de entendimento do que está sendo demonstrado.

A taxa média de acerto dos três voluntários que utilizaram a ferramenta foi de 35,13 % e daqueles que não utilizaram foi de 34,23 %.

3.2.5 Vulnerabilidades encontradas

Neste item foi utilizada uma máquina com serviços vulneráveis conhecidos e solicitado que os voluntários encontrassem as vulnerabilidades e listassem o código *Common Vulnerabilities and Exposure* (CVE) de cada uma delas. Não é possível afirmar com total certeza a quantidade de vulnerabilidades presentes no cenário de teste, pois podem haver vulnerabilidades ainda desconhecidas pela comunidade, por isso, para fins de comparação e referência, foi adotado como número máximo de vulnerabilidades as encontradas pelos voluntários com melhor performance, levando em conta que todas as vulnerabilidades encontradas pelos demais *pentesters* estão contidas nessas 93 listadas pelo voluntário com muito conhecimento que utilizou a ferramenta.

Tabela 08 – Vulnerabilidades Encontradas

	Vulnerabilidades encontradas	Comparação
Voluntário com **MUITO CONHECIMENTO** Utilizando a ferramenta SAPIENS	93	100%
Voluntário com **MUITO CONHECIMENTO** Sem o uso da ferramenta SAPIENS	39	41,93%
Voluntário com **POUCO CONHECIMENTO** Utilizando a ferramenta SAPIENS	47	50,53%
Voluntário com **POUCO CONHECIMENTO** Sem o uso da ferramenta SAPIENS	01	1,07%
Voluntário **SEM CONHECIMENTO** Utilizando a ferramenta SAPIENS	01	1,07%
Voluntário **SEM CONHECIMENTO** Sem o uso da ferramenta SAPIENS	04	4,3%

Fonte – O autor, 2016

Como é possível observar, a diferença entre o resultado alcançado pelo voluntário com MUITO CONHECIMENTO que não utilizou o SAPIENS (41.93%) para os outros dois com menos conhecimento e experiência que também não utilizaram a ferramenta (1,07% e 4,3%) é bastante expressiva. Com isso é possível inferir que a tarefa de identificar vulnerabilidades em uma máquina é uma tarefa que exige um grau elevado de conhecimento por parte do *pentester*.

Outro ponto que exige uma apreciação mais detalhada é a "vantagem" alcançada pelo voluntário com POUCO CONHECIMENTO que usou o SAPIENS para o com MUITO CONHECIMENTO que não usou a ferramenta. Mesmo com menos domínio das técnicas e procedimentos encontrou 08 vulnerabilidades a mais, reforçando os benefícios percebidos com o uso do software. Em contrapartida, o voluntário com POUCO CONHECIMENTO que usou o SAPIENS identificou apenas 01 vulnerabilidade, resultado bem abaixo dos outros dois usuários que lançaram mão da ferramenta.

Uma hipótese para o evento observado é a interação que o SAPIENS exige nessa fase do *pentest*. Por ocasião da identificação das vulnerabilidades é solicitado ao usuário que informe manualmente a versão do serviço que deseja encontrar falhas. Destarte, caso seja inserido um valor incompleto ou errado a ferramenta será incapaz de realiza a operação com sucesso. Torna-se evidente então a necessidade de um conhecimento mínimo por parte do *pentester* para encontrar vulnerabilidades em um *host*, mesmo utilizando uma ferramenta automatizada.

A taxa média de acerto dos três voluntários que utilizaram a ferramenta foi de 50,53 % e daqueles que não utilizaram foi de 15,76 %.

3.2.6 Outras considerações

Este item possibilitou ao voluntário relatar outros dados encontrados durante a auditoria de segurança realizada.

Tabela 08 – Considerações apresentadas pelos voluntários durante o *pentest*

	Considerações apresentadas
Voluntário com **MUITO CONHECIMENTO** Utilizando a ferramenta SAPIENS	- 04 números de telefone da empresa alvo - Bloco IP da empresa alvo - Porta UDP 68 aberta - Pastas indexadas do servidor web - Netbios:metasploitable - Unix 2.624-16 i686 (Ubuntu 8.04) -Host: Metasploitable.localdomain
Voluntário com **MUITO CONHECIMENTO** Sem o uso da ferramenta SAPIENS	- Netbios:metasploitable - Unix 2.624-16 i686 (Ubuntu 8.04) -Host: Metasploitable.localdomain
Voluntário com **POUCO CONHECIMENTO** Utilizando a ferramenta SAPIENS	- 04 números de telefone da empresa alvo - Bloco IP da empresa alvo - Porta UDP 68 aberta - Pastas indexadas do servidor web - Netbios:metasploitable - Unix 2.624-16 i686 (Ubuntu 8.04) -Host: Metasploitable.localdomain
Voluntário com **POUCO CONHECIMENTO** Sem o uso da ferramenta SAPIENS	Trata-se de uma máquina Linux 2.6.9-2.6.33, possui também o samba instalado, postgre sql e apache.
Voluntário **SEM CONHECIMENTO** Utilizando a ferramenta SAPIENS	- 04 números de telefone da empresa alvo - Bloco IP da empresa alvo
Voluntário **SEM CONHECIMENTO** Sem o uso da ferramenta SAPIENS	Nada relatado

Fonte – O autor, 2016

Este item torna-se bastante relevante, pois não foi oferecida uma "trilha" a ser seguida pelo voluntário como nas demais questões, onde era solicitado alcançar informações pontuais.

Os voluntários que utilizaram o SAPIENS alcançaram resultados superiores às suas respectivas duplas, obtendo mais informações sobre o domínio e a máquina alvo.

3.2.7 Duração do teste

Nesta questão solicitou-se aos voluntários que medissem o tempo dependido na realização da atividade.

Tabela 08 – Duração do teste

	Duração
Voluntário com **MUITO CONHECIMENTO** Utilizando a ferramenta SAPIENS	24 minutos
Voluntário com **MUITO CONHECIMENTO** Sem o uso da ferramenta SAPIENS	1 hora e 40 minutos
Voluntário com **POUCO CONHECIMENTO** Utilizando a ferramenta SAPIENS	24 minutos
Voluntário com **POUCO CONHECIMENTO** Sem o uso da ferramenta SAPIENS	54 minutos
Voluntário **SEM CONHECIMENTO** Utilizando a ferramenta SAPIENS	27 minutos
Voluntário **SEM CONHECIMENTO** Sem o uso da ferramenta SAPIENS	40 minutos

Fonte – O autor, 2016

Com base nos dados coletados é possível perceber que o tempo médio gasto por aqueles que utilizaram a ferramenta foi de 25 minutos, enquanto os que realizaram o *pentest* "a mãos livres" foi de 1 hora e 4 minutos. Nota-se portanto que a ferramenta garante uma velocidade 4 vezes maior à atividade.

3.3. ANÁLISE DOS RESULTADOS

A análise das respostas oferecidas no "Questionário de percepção das funcionalidades necessárias em uma ferramenta automatizada de Testes de Penetração" revelou, com base na população ouvida, que em mais da metade (59.1%) das organizações militares não foi realizado qualquer tipo de auditoria de segurança no último ano, sendo que os fatores determinantes para essa baixa frequência são: receio de expor negativamente o nome da instituição (73.9%); falta de preocupação com segurança (60.9%); e falta de profissionais capacitados para o ofício (56.5%). Uma alternativa aceita pela maioria (82.6%) como solução para aumentar a frequência destes testes é a utilização de uma ferramenta automatizada que possua as seguintes características: relatório detalhado e com sugestões de correção (82.6%); escalabilidade, ou seja, permitir a integração com novos scripts ou funcionalidades desenvolvidas pelos usuários (60.9%); Segurança dos dados obtidos e do canal por onde essas informações trafegam (47.8%); e Integração com outras ferramentas, como Nessus, Metasploit e GFI Languard (43.5%).

Ao analisar a efetividade da ferramenta SAPIENS através do "Questionário de percepção de uso da ferramenta SAPIENS", observou-se que os voluntários que utilizaram o

software apresentaram percentual superior aos que empregaram outros meios nos seguintes aspectos: identificação de servidores ativos (média 65,39% superior), identificação de e-mails de funcionários (média 35,45% superior) e identificação de vulnerabilidades conhecidas (média 34,77% superior). Além disso, o tempo de execução do *pentest* com a ferramenta foi em média 4 vezes menor. Em contrapartida o SAPIENS obteve rendimento inferior no quesito "nomes de funcionários", com percentual 58,34% menor.

CONCLUSÃO

A presente pesquisa buscou responder o seguinte problema: a automação das etapas de um teste de penetração garante maior eficiência e eficácia ao ofício do *pentester*? Para responder este questionamento foram conduzidas atividades que permitiram alcançar uma resposta precisa, com base em dados objetivos.

Antes de confrontar o questionamento supracitado, julgou-se necessário entender com mais profundidade alguns fatores envolvidos com a execução de auditorias de segurança. Neste ponto, ficou evidente que, mesmo conscientes dos benefícios de um teste de penetração, essa atividade ainda é pouco executada pelas organizações por motivos que variam desde a falta de preocupação com segurança até a falta de profissionais capacitados para o ofício.

Analisando estes dados, percebeu-se que a falta de profissionais capacitados, uma das condições listadas como causa do problema, poderia ser minimizada através do uso de uma ferramenta automatizada. Essa hipótese ganhou força quando a maioria dos profissionais ouvidos durante a pesquisa julgaram que o uso de uma ferramenta automatizada traria um incremento na frequência de *pentests* realizados em suas organizações.

Tendo este cenário como pano de fundo, foi desenvolvida uma ferramenta para esclarecer o problema que norteou a pesquisa. Para tanto, avaliou-se os resultados obtidos por *pentesters* com níveis de conhecimento diversificados em um ambiente de teste com vistas a avaliar se os resultados obtidos pelos profissionais que lançaram mão da ferramenta seriam melhores do que os alcançados por aqueles que não usaram-na.

A análise dos resultados revelou que o emprego de um software automatizado garante efeitos melhores por ocasião do *pentest*, confirmando as seguintes hipóteses apresentadas na introdução deste trabalho: a utilização de uma ferramenta automatizada torna as ações de um teste de penetração mais rápidas devido ao maior poder de processamento do software em comparação com os *pentesters* (H1); e o uso de uma ferramenta automatizada impõe uma série de testes pré-definidos a serem executados, garantindo uma cobertura mais completa, quando comparado com a ação de um profissional de segurança da informação, de todas as eventuais vulnerabilidades presentes em uma rede alvo. Dessa maneira, eficácia de testes que lançam mão de softwares dessa natureza é maior do que os demais. (H3).

Outrossim, percebeu-se que mesmo entre profissionais com pouco conhecimento e experiência em ações de *pentest* a ferramenta garantiu resultados consistentes e além do que seria esperado para o perfil, pois era de se supor que devido ao alto grau de tecnicidade

envolvendo auditorias de segurança esses voluntários alcançassem um rendimento bastante abaixo dos profissionais com mais experiência.

Além de alcançar uma resposta ao problema apresentado, o presente trabalho oferece como contribuição uma nova ferramenta, cujos detalhes estão apresentados no apêndice “C”, que pode colaborar com organizações que, mesmo carentes de profissionais especialistas em segurança da informação, desejam elevar o nível de segurança de sua infraestrutura lógica. Como revelado durante a discussão do referencial teórico, o mercado goza de outras ferramentas com escopos similares, todavia, diferente das demais, o SAPIENS é gratuito, possui código fonte aberto, segue uma metodologia com reconhecimento e qualidade internacional (ISSAF) e vai ao encontro das demandas reveladas por profissionais da área de segurança.

Ao longo do desenvolvimento da pesquisa surgiram outras possibilidades que extrapolavam os limites impostos na delimitação do tema. Destarte, sugere-se como possibilidade de trabalhos futuros a inserção de novas funcionalidades do SAPIESN, como utilização de banco de dados, compatibilidade com softwares de ataque como METASPLOIT e desenvolvimento de uma interface web mais amigável por ocasião da apresentação do relatório. Outra temática sugerida é a comparação do desempenho do SAPIENS com softwares que compartilham a mesma finalidade, como NESSUS e GFI Languard.

Mesmo diante dos resultados alcançados é prematuro afirmar que a automação substitui o fator humano em ações relacionadas a auditorias de segurança. O que se constatou foi uma aparente vantagem percebida em um cenário particular, cabendo testes mais diversificados e com cenários mais robustos para mensurar se o melhor caminho a ser seguido pelas organizações é investir em ferramentas em detrimento da capacitação dos recursos humanos. Contudo a resposta alcançada por este estudo revelou que uma boa ferramenta de automação pode ter uma acurácia mais refinada, quando comparado à profissionais que lançam mão apenas do seus próprios recursos.

REFERÊNCIAS

ABOUT KALI LINUX**.** Disponível em: https://www.kali.org/about-us/ Acesso em: 21 mar 2016

BORGERS, Luiz E. **Python para Desenvolvedores**. 1. ed. São Paulo-SP. Editora: Novatec, 2015

BRASIL. Estado-Maior do Exército Brasileiro. Portaria Nº 185: Cria o Curso de Guerra Cibernética para Sargentos e estabelece suas condições de funcionamento. Brasília, 2012.

HERZOG, P. **OSSTMM 3** – The Open Source Security Testing Methodology Manual. Barcelona, Espanha: ISECOM, 2010.

NESSUS. Disponível em: https://www.tenable.com/products/nessus-vulnerability-scanner. Acessado em: 21 mar 16

NEVES, Julio Cezar. **Linux: programação shell**. 2. ed. Rio de Janeiro: Brasport, 2001.

NMAP. Disponível em: https://nmap.org/ acessado em 11/02/16

_____. Disponível em: https://nmap.org/book/nse-usage.html#nse-categories acessado em 11/02/16

Penetration Testing Execution Standard. Disponível em: http://www.pentest-standard.org/index.php/PTES_Technical_Guidelines. Acessado em: 01 de julho de 2015

RAMOS J. J. A. **Sistema Automático para Realização de Testes de Penetração.** Instituto Politécnico de Beja, 2013.

RATHORE B. **Information System Security Assessment Framework (ISSAF**) Draft 0.2.1B, 2006.

SCARFONE, K. et al. **Technical Guide to Information Security Testing and Assessment**. Gaithersburg: NIST, 2008.

APÊNDICE "A" -QUESTIONÁRIO DE PERCEPÇÃO DAS FUNCIONALIDADES NECESSÁRIAS EM UMA FERRAMENTA AUTOMATIZADA DE TESTES DE PENETRAÇÃO

Pretende-se, por meio deste questionário, identificar funcionalidades úteis em uma ferramenta automatizada de Testes de Penetração.
Este instrumento visa à coleta de dados referente ao trabalho de Conclusão do Curso de Pós Graduação em Redes de computadores com ênfase em segurança do aluno Felipe Rodrigues de Vasconcellos, cujo escopo é desenvolver uma ferramenta para tornar as tarefas de um teste de penetração mais céleres e metodológicas, garantindo uma eficiência mais refinada através da diminuição do fator humano nas ações de descoberta e exploração de vulnerabilidades em sistemas informáticos.
Buscando-se sugerir propostas que atendam ao objetivo da pesquisa, o senhor foi selecionado pelo desempenho de funções relacionadas à segurança da informação para responder a este questionário.
O Centro Universitário de Brasília e seus alunos pesquisadores agradecem sua valiosa contribuição.
*Obrigatório

Qual o nome do senhor

(resposta opcional)

Quantos anos de experiência na área de segurança o senhor possui? *

- ○ Menos de 1 ano
- ○ Mais de 1 e menos de 3 anos
- ○ Mais de 3 e menos de 5 anos
- ○ Mais de 5 anos

De acordo com a experiência do senhor na área de segurança, quais fatores inibem a realização de testes de penetração nas empresas e instituições?
Marque quantas alternativas julgar necessário

- ☐ Caso o senhor acredite que os testes de penetração são atividades popularizadas e largamente empregadas em empresas e organizações, MARQUE APENAS ESSA OPÇÃO.

- ☐ Falta de profissionais capacitados para o ofício
- ☐ Custo financeiro da atividade
- ☐ Falta de preocupação com segurança
- ☐ Ineficiência desse tipo de teste
- ☐ Desconhecimento desse tipo de teste
- ☐ Outras medidas de segurança substituem o pentest
- ☐ Receio de expor negativamente o nome da instituição
- ☐ Outro: _____

Em uma ferramenta de automação de testes de penetração, escolha as 4 (QUATRO) características que o senhor julga mais relevante nesse tipo de de software. *
(SELECIONE APENAS 4 OPÇÕES)

- ☐ Interface Gráfica amigável
- ☐ Relatório detalhado e com sugestões de correção
- ☐ Possibilidade de trabalho colaborativo entre dois ou mais profissionais simultaneamente
- ☐ Possibilidade de executar a ferramenta remotamente, sem a necessidade de estar na rede alvo
- ☐ Integração com outras ferramentas, como Nessus, Metasploit e GFI Languard.
- ☐ Segurança dos dados obtidos e do canal por onde essas informações trafegam
- ☐ Possibilidade de executar ações com o mínimo de interação do usuário, possibilitando a realização do pentest por empresas e profissionais sem tantos recursos e/ou conhecimento técnico
- ☐ Garantir anonimato na ações desenvolvidas
- ☐ Escalabilidade, ou seja, permitir a integração com novos scripts ou funcionalidades desenvolvidas pelos usuários
- ☐ Outro: _____

O órgão/empresa que o senhor trabalha realiza auditorias de segurança preventivas ao menos uma vez por ano?

- ○ Sim, contrata profissionais externos para realizar esse tipo de atividade.
- ○ Sim, é conduzida exclusivamente por funcionários internos
- ○ Não, durante o período que estou na empresa nunca foi realizada nenhuma avaliação dessa natureza

- Não, são realizadas algumas auditorias, mas com baixa frequência (menos de uma vez por ano)
- Desconheço

O senhor acredita que, caso sua empresa/instituição possuisse uma ferramenta automatizada para realizar testes de penetração, esse tipo de atividade seria realizada com mais frequência para mitigar eventuais vulnerabilidades nos sistemas computacionais?

- Sim
- Não
- Indiferente
- Minha empresa/instituição já possui uma ferramenta dessa natureza
- Outro:

O senhor tem outra consideração ou sugestão sobre o assunto em tela?

_____
_____
_____

APÊNDICE "B" - QUESTIONÁRIO DE PERCEPÇÃO DE USO DA FERRAMENTA SAPIENS

Pretende-se, por meio deste questionário, identificar o comportamento da ferramenta Sapiens na automação de Testes de Penetração, bem como aferir os eventuais benefícios e desvantagens trazidas pelo seu uso.

Este instrumento visa à coleta de dados referente ao trabalho de Conclusão do Curso de Pós Graduação em Redes de computadores com ênfase em segurança do aluno Felipe Rodrigues de Vasconcellos, cujo escopo é desenvolver uma ferramenta para tornar as tarefas de um teste de penetração mais céleres e metodológicas, garantindo uma eficiência mais refinada através da diminuição do fator humano nas ações de descoberta e exploração de vulnerabilidades em sistemas informáticos.

Buscando-se sugerir propostas que atendam ao objetivo da pesquisa, o senhor foi selecionado pelo desempenho de funções relacionadas à segurança da informação para responder a este questionário.

O Centro Universitário de Brasília e seus alunos pesquisadores agradecem sua valiosa contribuição.

a. Sobre o pentest

Questão 01. Qual(is) servidor(es) da rede alvo o senhor encontrou? Liste o(s) endereço(s) IP encontrado(s):

_____
_____
_____

Questão 02. Quantos nome(s) de funcionário(s) o senhor encontrou? Liste o(s) nome(s) encontrado(s):

_____
_____
_____

Questão 03. Quantos e-mails relacionados ao domínio alvo (ex: @dominio.com) o senhor encontrou? Liste os e-mail encontrados:

_____
_____
_____

Questão 04. Qual(is) máquina(s) da rede alvo está(ão) ativa(s). Liste o(s) endereço(s) IP encontrado(s):

_____
_____
_____

Questão 05. Qual(is) vulnerabilidade(s) encontrada(s) no endereço IP alvo? Cite o número do CVE ou uma rápida descrição da(s) vulnerabilidade(s) encontrada(s).

_____
_____
_____
_____

Questão 06. O senhor encontrou alguma outra informação útil sobre o domínio e/ou máquina alvo?

_____
_____
_____
_____

APÊNDICE "C" – CÓDIGO FONTE DO SOFTWARE SAPIENS

```
#!/usr/bin/env python
#coding: utf8

#
=================================================================
===========
#
# Trabalho de Conclusão de Curso - Pós de Redes com ênfase em Segurança
#
# Autor: Felipe Rodrigues de Vasconcellos
# Orientador: Javier Obadia
#
# Ferramenta : SAPIENS
# Sugestões entre em contato: aspvasconcellos@gmail.com
#
#
=================================================================
===========

# Importando bibliotecas
# bibliotecas necessárias para utilizar o selenium
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.support.ui import Select
from selenium.common.exceptions import NoSuchElementException
from selenium.common.exceptions import NoAlertPresentException
# bibliotecas necessárias para executar comandos em Shell
import unittest, time, re
import os
import commands


# Criando um Alias que permitirá executar comandos em shell apenas digitando a palavra
"comando" antes do comando em shell
comando = os.system

#antes de rodar o programa apague o diretório PENTEST caso exista algum e os arquivos
REALTÓRIO e INTERMEDIARIO
comando('rm -r /var/www/html/pentest 2> /dev/null 1> /dev/null')
comando('rm /var/www/html/sapiens.html 2> /dev/null')

def inicio():
        global dominioInformado
        global relatorio
```

```
        global diretorioUso
        global intermediario
        global ipInformado
        global horaInicio
        comando('clear')
        print "\nPara uma experiência melhor com o SAPIENS, maximize a janela do
terminal."
        print "\nApós a execução da ferramenta acesse no browser \"localhost/sapiens.html\"
para visualizar o relatório gerado.\n\n"
        respInicio = raw_input('Para iniciar o pentest digite "INICIAR"\n\n>>> ')
        # Cria um loop que, enquanto o usuário não digitar "iniciar" o programa não
prossegue.
        while respInicio != "iniciar" and respInicio != "Iniciar" and respInicio != "INICIAR":
                respInicio = raw_input('\nDigite INICIAR para prosseguir.\n\n>>>')
        #banner inicial quando a ferramenta é executada
        comando('clear')
        print " "
        print " "
        print " "
        print " "
        print
"#############################################################################
################ "
        print
"#############################################################################
################ "
        print "
          "
        print "    //   ) )     // | |     //   ) ) )       / /        //   / /     /|    / /     //   ) ) )  "
        print "   ((          //__| |    //___/ /       / /        //____       //|   / /     ((         "
        print "     \\        / ___  |   / _____ /       / /        / _____       // |  / /       \\       "
        print "       ) )    //   | |  //            / /        //           //  | / /          ) )    "
        print "((___ / /    //    | | //           __/ /___     //____/ /    //   |/ /     ((___ / /     "
        print "
          "
        print
"#############################################################################
################  "
        print
"#############################################################################
################  "

        # Cria a variável "horaInicio" cujo conteúdo será o dia e a hora de início do uso da
ferrametna "sapiens".
        horaInicio = commands.getoutput('date')
```

```
	# Criação da variável "dominioInformado" que será o alvo dos comandos executados pela ferramenta
	dominioInformado = raw_input('\nInforme o Dominio alvo:\n\n>>> ')

	# Cria a variável "ipInformado" com base no ping executado contra o servidor web
	# do "dominioInformado".
	ipInformado = commands.getoutput('ping -c 1 www.%s | cut -s -d "(" -f2 | cut -d ")" -f1 | head -1' % dominioInformado)
	# Cria o diretório e arquivo onde os arquivos de relatório e auxiliares ficarão
	comando('mkdir /var/www/html/pentest')
	# Cria a variável "diretorioUso" que contém o diretório onde a ferramenta salvará os arquivos gerados
	diretorioUso = str(commands.getoutput('echo "/var/www/html/pentest"'));
	# Cria os arquivos "sapiens.html" (relatório final gerado pela ferramenta) e "intermediario" (arquivo de apoio com saídas de comando), que	será
	# excluído ao final da execução da ferramenta
	comando('touch /var/www/html/sapiens.html %s/intermediario' % diretorioUso)
	# Cria a variavel "relatorio" que será o destino final das informações filtradas e obtidas pela ferramenta
	relatorio = str(commands.getoutput('echo "/var/www/html/sapiens.html"'));
	# Cria a variavel "intermediario" que será o destino das informações não filtradas e obtidas pela ferramenta
	intermediario = str(commands.getoutput('echo "%s/intermediario"' % diretorioUso));
	# Inicia o apache
	comando('service apache2 start')

	# instala pacotes necessarios "httrack", "sipcalc" e "ipcalc"
def preparar_ambiente():
	comando('apt-get -y update')
	comando('apt-get -y install sipcalc ipcalc')
	comando('apt-get -y install httrack')


# Cria variável pagina, que abre o arquivo sapiens.html com opção append, ou seja, tudo que for escrito no arquivo será adicionado ao fim do mesmo.
# nesse ponto a página html, que será o relatório gerado pela ferramenta, começa a ser contruída. É inserido seu header e algumas tags do body.
def geraHTML():
	pagina=open("/var/www/html/sapiens.html","a")
	pagina.write("""
	<!DOCTYPE html>
	<html>		<head>
	<title>Sapiens</title>
	<meta charset="utf-8"/> <meta name="description" content="" />
	<link rel="stylesheet" href="style.css" type="text/css" />
	</head>
	<body>
```

```
        <div id="page">
        <div id="logo">
        <h1><a href="/" id="logoLink">Sapiens</a></h1>
        </div>
        <div id="nav">
        <ul>
        <li><a href="#/issaf.pdf">ISSAF</a></li>
        <li><a href="#/pentest">Docs</a></li>
        <li><a href="#/contato.html">Contato</a></li>
        </ul>
        </div>
        <div id="content">
        <h2>Relatório de Pentest</h2>
        <p>Relatório produzido pela ferramenta SAPIENS. Esse documento é produto de uma auditoria de segurança que utilizou a metodologia proposta pela "Information Systems Security Assessment Framework (ISSAF)". Para detalhar cada uma das etapas realizadas acesse o documento da ISSAF disponibilizado no link acima.</p>
        <p>Ao longo da auditoria foram produzidos arquivos intermediários com endereços IP de servidores e outros dados. Caso deseje analisar esses documentos acesse o link Docs.</p>
        """)
        pagina.close()

# Início das ações de pentest executadas pela ferramenta
# Cada uma dessas ações está contida em uma função nomeada por uma letra, seguida por três números (Ex: A_1_1_2)
# Essa nomeclatura é uma referência a metodologia de pentest sugerida pela ISSAF. Com isso é possível comparar
# as ações sugeridas pela ISSAF em cada umas das fases do pentest e o que foi implementado na ferramenta.

################################################################################
################################################################################
###
################################################
###     1.1.2 Examine Domain Name System / Find Out Domain Registration Info and IP Block Owned  ################################################
###                                     ##### PASSO 01 ####
         ################################################
################################################################################
################################################################################
def A_1_1_2():
        # Limpa a tela do usuário
        comando('clear')
        # Executa o banner explicando o passo 01
        banner_A_1_1_2()
        # Programa fica em espera por 7 segundos para o usuário ler o banner do passo 01
```

```
        comando('sleep 7')
        global blocoIP
        # Insere linhas com "#" no "relatorio"
        comando('echo " PASSO 01 - Dados coletados dos registros de domínio e de blocos IP (A 1.1.2 do ISSAF) <br />" >> %s' % relatorio)
        comando('echo "##########################################################################################################<br />" >> %s' % relatorio)
        # Executa o comando whois e envia o output para o arquivo intermediário, para que as informações úteis sejam filtradas
        comando('whois %s > %s 2> /dev/null' % (dominioInformado, intermediario))
        # Filtra do arquivo "intermediario" a Organização do domínio alvo e insere o resultado no arquivo "relatorio"
        comando('echo "<p><h3>ORGANIZAÇÃO</h3></p>" >> %s' % relatorio)
        comando('cat %s | egrep \'(owner:|Organization)\' | cut -d: -f2 | tr -s " " | tr -d ^" " | uniq >> %s' % (intermediario, relatorio))
        # Filtra do arquivo "intermediario" o responsável pelo domínio alvo e insere o resultado no arquivo "relatorio"
        comando('echo "<p></p><p><h3>RESPONSÁVEL PELO DOMÍNIO</h3></p>" >> %s' % relatorio)
        comando('cat %s | egrep \'(responsible|Registrant Name)\' | cut -d: -f2 | tr -s " " | tr -d ^" " >> %s' % (intermediario, relatorio))
#       # Filtra do arquivo "intermediario" os servidores DNS do domínio alvo e insere o resultado no arquivo "relatorio"
#       comando('echo "<p></p><p><h3>DNS</h3></p>" >> %s' % relatorio)
#       comando('cat %s | egrep \'(nserver|Name Server)\' | cut -d: -f2 | tr -d " " | sort -u | uniq -i | sed \'s-$-<p></p>-g\' >> %s' % (intermediario, relatorio))
        # Cria um arquivo temporario com nomes de servidores DNS que será utilizado nas funções A_1_1_3 e A_1_1_23
        comando('cat %s | egrep \'(nserver|Name Server)\' | cut -d " " -f 6 >> %s/nomeDns 2> /dev/null' % (intermediario, diretorioUso))
        # Cria o arquivo temporario com IP DNS que será utilizado nas funções A_1_1_3 e B_1_1_28
        comando('host `sed -n 1p /var/www/html/pentest/nomeDns` 2> /dev/null | egrep -o \'([0-9]{1,3}\.){3}[0-9]{1,3}\' > %s/ipDns' % diretorioUso)
        comando('host `sed -n 2p %s/nomeDns` 2> /dev/null | egrep -o \'([0-9]{1,3}\.){3}[0-9]{1,3}\' >> %s/ipDns' % (diretorioUso, diretorioUso))
        comando('host `sed -n 3p %s/nomeDns` 2> /dev/null | egrep -o \'([0-9]{1,3}\.){3}[0-9]{1,3}\' >> %s/ipDns' % (diretorioUso, diretorioUso))
        comando('host `sed -n 4p %s/nomeDns` 2> /dev/null | egrep -o \'([0-9]{1,3}\.){3}[0-9]{1,3}\' >> %s/ipDns' % (diretorioUso, diretorioUso))
        # Filtra do arquivo "intermediario" o EMAIL do responsável pelo domínio alvo
        comando('echo "<p></p><p><h3>EMAIL</h3></p>" >> %s' % relatorio)
        comando('cat %s | egrep \'(e-mail:|Email:)\' | tr -d " " | sort -u | uniq -i | sed \'s-$-<p></p>-g\' | egrep \'mail\'>> %s' % (intermediario, relatorio))
```

```
        # Executa o comando whois contra o IP do servidor web do domínio informado e redirecionando a saída para o arquivo "relatório".
        # com isso é possível obter a bloco IP, o qual esse servidor web pertence
        comando('whois %s > %s' % (ipInformado, intermediario))
        # Filtra do arquivo "intermediario" o EMAIL do responsável pelo domínio alvo e insere o resultado no arquivo "relatorio"
        comando('echo "<p></p><p><h3>BLOCO IP</h3></p>" >> %s' % relatorio)
        comando('cat %s | egrep \'(inetnum:|NetRange)\' | cut -d: -f2 | tr -s " " | tr -d ^" " | sed \'s-$-<p></p>-g\' >> %s' % (intermediario, relatorio))
        # Filtra do arquivo "intermediario" o CNPJ responsável pelo domínio alvo e insere o resultado no arquivo "relatorio"
        comando('echo "<p></p><p><h3>CNPJ</h3></p>" >> %s' % relatorio)
        comando('cat %s | egrep ownerid: | cut -d: -f2 | tr -d " " >> %s' % (intermediario, relatorio))
        # Filtra do arquivo "intermediario" o FAX ou TELEFONE do responsável pelo domínio alvo e insere o resultado no arquivo "relatorio"
        comando('echo "<p></p><p><h3>FAX/PHONE</h3></p>" >> %s' % relatorio)
        comando('cat %s | egrep \'(Fax|Phone)\' | cut -d: -f2 | sed \'s-$-<p></p>-g\' >> %s' % (intermediario, relatorio))
        # Filtra do arquivo "intermediario" nomes(s) de responsável(is) pelo domínio alvo e insere o resultado no arquivo "relatorio"
        comando('echo "<p></p><p><h3>POSSÍVEIS NOMES DE FUNCIONÁRIOS</h3></p>" >> %s' % relatorio)
        comando('cat %s | egrep \'(person:|Tech Name:|Registrant Name|Admin Name:)\' | cut -d: -f2 | uniq -i | sed \'s/^ */\' | sed \'s-$-<p></p>-g\' >> %s' % (intermediario, relatorio))
        # Preparação para listar o range de IP do alvo com base no IP do servidor web do domínio informado
        # Cria a variável "blocoIp" que é o campo "inetnum ou CIDR" do comando whois
        blocoIP = commands.getoutput('cat %s | egrep \'(inetnum:|CIDR:)\' | cut -d: -f2 | tr -s " " | tr -d ^" " | sed -n 1p' % intermediario)
        # O output do comando Whois é diferente para cada RIR. Assim, se o domínio for .br terá um output diferente de um .com. A variável "nic" e o
        # laço IF a seguir tratam essa diferença. Caso o valor de NIC seja "normal" o IF  a seguir não terá nenhum efeito. Caso seja "diferente"
        # o IF a seguir mudará o valor da variável "blocoIP"
        nic = commands.getoutput('if echo %s | egrep \'/\' 1> /dev/null 2> /dev/null; then z="normal"; else z="diferente"; fi; echo $z' % blocoIP)
        if nic == "diferente":
                blocoIP = commands.getoutput('ipcalc %s | sed -n \'2p\" % blocoIP)
        # CALCULO BLOCO IP - o bloco IP é calculado e com o mando IPCALC descobre-se o primeiro e último IP do range. Com base nesses é criada uma # lista de IPs chamada range.txt. O arquivo range.txt será usado na função "B_1_1_28", quando serão analisados quais destes IP estão ativos.
        comando('ipcalc %s | egrep \'HostM\' | egrep -o \'([0-9]{1,3}\.){3}[0-9]{1,3}\' > %s/inter.txt' % (blocoIP, diretorioUso))
        comando('a=`cat %s/inter.txt | cut -d "." -f 4 | head -1`; b=`cat %s/inter.txt | cut -d "." -f 4 | tail -n 1`; c=`cat %s/inter.txt | cut -d "." -f 1,2,3 | uniq`; for i in $(seq $a $b); do echo
```

```
"$c.$i" >> %s/inter2.txt; done; cat %s/inter2.txt | sort | uniq > %s/range.txt; rm %s/inter*.txt'
% (diretorioUso, diretorioUso, diretorioUso, diretorioUso, diretorioUso, diretorioUso,
diretorioUso))
        # Idenfifica o bloco IP com base no comando ipcalc e insere o resultado no arquivo
"relatorio"
        comando('echo "<p></p><p><h3>BLOCOS IP</h3></p>" >> %s' % relatorio)
        comando('echo "<p></p><p>O IP do endereço web alvo é propriedade de uma
organização que possui o(s) seguinte(s) range(s):</p>" >> %s' % relatorio)
        comando('cat %s | egrep \'(inetnum:|CIDR:)\' | cut -d: -f2 | tr -s " " | tr -d ^" " | sed \'s-
$-<p></p>-g\' >> %s' % (intermediario, relatorio))
        # Adiciona no arquivo sapiens.html as contramedidas para as ações 1.1.2
        pagina=open("/var/www/html/sapiens.html","a")
        pagina.write("""
        <p><h4>Ferramentas utilizadas: WHOIS.</h4></p><p><h4>
        <p><h4>Contramedidas:</h4></p><p><h4> 1 - utilize nomes genéricos para as
funções. Use generic role names (like postmaster)</h4></p>
        <p><h4>2 - use endereços de email genéricos para as funções (exemplo
postmaster@companhia.com)<h4></p>
        <p><h4>3 - utilize apenas 01 contato telefônico, que normalmente não é de
conhecimento público. Caso receba chamdas nesse número saberá onde ele foi encontrado.
Alternativamente, use números de outra localidade geográfica (outro DDI ou
DDD)</h4></p>
""")
        pagina.close()
        # Insere linhas com "#" no "relatorio"

##############################################################################
##############################################################################
###
################################################
###     1.1.3 Examine Domain Name System - Check for the Authoritative Name Server
                ################################################
###                             ##### PASSO 02 ####
         ################################################
##############################################################################
##############################################################################
def A_1_1_3():
        # Limpa a tela do usuário
        comando('clear')
        # Executa o banner explicando o passo 02
        banner_A_1_1_3()
        # Deixa o programa parado por 5 segundos, assim o usuário consegue ler o banner
        comando('sleep 5')
        # Insere linhas com "#" no "relatorio"
        comando('echo " PASSO 02 - Dados obtidos de consultas DNS (A 1.1.3 do
ISSAF)<br />" >> %s' % relatorio)
```

```
        comando('echo
"##############################################################################
############################################################<br />" >> %s' %
relatorio)
        # Através do comando "dig", obtém todos os registros de DNS do domínio informado
e redireciona a saída para o arquivo "intermediario"
        comando('dig -t any %s >> %s' % (dominioInformado, intermediario))
        comando('dig %s MX >> %s' % (dominioInformado, intermediario))
        comando('dig %s NS >> %s' % (dominioInformado, intermediario))
        comando('dig %s A >> %s' % (dominioInformado, intermediario))
        # Filtra do arquivo "intermediario" os servidores DNS do domínio alvo e insere o
resultado no arquivo "relatorio"
        comando('echo "<p></p><p><h3>SERVIDOR DNS</h3></p>" >> %s' % relatorio)
        comando('cat %s | egrep \'NS\' | cut -d " " -f2 | sort -u | cut -d "S" -f 2 | egrep
\'(com.|br.)\' | cut -c 2- | sort -u | uniq -i >> %s/nomeDns' % (intermediario, diretorioUso))
        comando('for i in $(seq 1 `wc -l %s/nomeDns | egrep -o [0-9]`); do ping -c 1 `sed -n
$i\'p\' %s/nomeDns` | egrep -o \'([0-9]{1,3}\.){3}[0-9]{1,3}\' | sort | uniq >> %s/ipDns 2>
/dev/null; done' % (diretorioUso, diretorioUso, diretorioUso))
        comando('for i in $(seq 1 `wc -l %s/nomeDns | egrep -o [0-9]`); do sed -n $i\'p\'
%s/nomeDns >> %s; echo "-->" >> %s; sed -n $i\'p\' %s/ipDns >> %s; echo "<p></p>" >>
%s;  done' % (diretorioUso, diretorioUso, relatorio, relatorio, diretorioUso, relatorio,
relatorio))
        servidoresDns = commands.getoutput('cat %s | grep \'NS\' | cut -d " " -f2 | sort -u | cut
-d "S" -f 2 | grep "com" | cut -c 2- 1> /dev/null'  % intermediario)
        # Filtra os servidores de email encontrados com o comando "dig" e insere o resultado
no arquivo "relatorio"
        comando('echo "<p></p><p><h3>SERVIDOR EMAIL</h3></p>" >> %s' %
relatorio)
        comando('cat %s | egrep \'(MX.*com.|MX.*br.)\' | cut -d " " -f2 | sort | uniq >>
%s/nomeMx' % (intermediario, diretorioUso))
        comando('for i in $(seq 1 `wc -l %s/nomeMx | egrep -o [0-9]`); do ping -c 1 `sed -n
$i\'p\' %s/nomeMx` | egrep -o \'([0-9]{1,3}\.){3}[0-9]{1,3}\' | sort | uniq >> %s/ipMx 2>
/dev/null; done' % (diretorioUso, diretorioUso, diretorioUso))
        comando('for i in $(seq 1 `wc -l %s/nomeMx | egrep -o [0-9]`); do sed -n $i\'p\'
%s/nomeMx >> %s; echo "-->" >> %s; sed -n $i\'p\' %s/ipMx >> %s; echo "<p></p>" >>
%s;  done' % (diretorioUso, diretorioUso, relatorio, relatorio, diretorioUso, relatorio,
relatorio))
        # Filtra outros IP encontrados com o comando "dig" e insere o resultado no arquivo
"relatorio"
        comando('echo "<p></p><p><h3>OUTROS IP</h3></p>" >> %s' % relatorio)
        comando('cat %s | grep \'A\' | egrep -o \'([0-9]{1,3}\.){3}[0-9]{1,3}\' | sort -u | uniq |
sed \'s-$-<p></p>-g\' >> %s' % (intermediario, relatorio))
        # Adiciona no arquivo sapiens.html as contramedidas para as ações 1.1.3
        pagina=open("/var/www/html/sapiens.html","a")
        pagina.write("""
        <p><h4>Ferramentas utilizadas: DIG.</h4></p><p><h4>
```

```
        <p><h4>Contramedidas:</h4></p><p><h4> 1 - Não há contramedidas para evitar consultas DNS, esse é um requisito para o correto funcionamento da internet;</h4></p>
        <p><h4>2 - Implementar restrição de consulta à versão do servidor DNS, bem como restrição a AXFR;<h4></p>
        <p><h4>3 - Nenhuma informação dos sistemas internos de uma organização devem ser disponibilizados nos servidores DNS externos.</h4></p>
""")
        pagina.close()



################################################################################
################################################################################
###
################################################
###     1.1.5 Examine Domain Name System - Check Spam/Attackers databases lookup
                ################################################
###                                     ##### PASSO 03 ####
         ################################################
################################################################################
################################################################################
def A_1_1_5():
        # Limpa a tela do usuário
        comando('clear')
        # Executa o banner explicando o passo 03
        banner_A_1_1_5()
        # Deixa o programa parado por 5 segundos, assim o usuário consegue ler o banner
        comando('sleep 5')
        global enderecoMx
        # Insere linhas com "#" no "relatorio"
        comando('echo " PASSO 03 - Verficação se o domínio e servidores de email constam em "blacklists" (A 1.1.5 do ISSAF)<br />" >> %s' % relatorio)
        comando('echo "################################################################################
################################################################<br />" >> %s' % relatorio)
        # Checa se os IP dos MX está em blacklists, utilizando o site "http://www.spamhaus.org"
        # e cria o arquivo enderecoMx, com os IP dos servidores de email, que será usado nas funções "A_1_1_18" e "B_1_1_28"
        comando('echo "<p></p><p><h3>VERIFICAÇÃO SE OS MX ESTÃO EM BLACKLIST</h3></p>" >> %s' % relatorio)
        comando('ipMx=$(dig +short -t mx %s | cut -d " " -f 2 > %s/mx.txt; ping -c 1 `cat %s/mx.txt | sed -n 1p` | egrep -o \'([0-9]{1,3}\.){3}[0-9]{1,3}\' | uniq) 2> /dev/null; echo "$ipMx  " >> %s; echo "<a href=\"http://www.spamhaus.org/query/ip/$ipMx\" target=\"_blank\"> -> [CLIQUE NO LINK] de verificação</a><p></p>" >> %s; echo
```

```
$ipMx >> %s/enderecoMx' % (dominioInformado, diretorioUso, diretorioUso, relatorio, relatorio, diretorioUso))
        comando('ipMx=$(dig +short -t mx %s | cut -d " " -f 2 > %s/mx.txt; ping -c 1 `cat %s/mx.txt | sed -n 2p` | egrep -o \'([0-9]{1,3}\.){3}[0-9]{1,3}\' | uniq) 2> /dev/null; echo "$ipMx  " >> %s; echo "<a href=\"http://www.spamhaus.org/query/ip/$ipMx\" target=\"_blank\"> -> [CLIQUE NO LINK] de verificação</a><p></p>" >> %s; echo $ipMx >> %s/enderecoMx' % (dominioInformado, diretorioUso, diretorioUso, relatorio, relatorio, diretorioUso))
        # Checa se o domínio alvo está em blacklists, utilizando o site "http://www.spamhaus.org"
        comando('echo "<p></p><p><h3>VERIFICAÇÃO SE O DOMINIO ESTÁ EM BLACKLIST</h3></p>" >> %s' % relatorio)
        comando('echo "Domínio: %s ->  <a href=\"http://www.spamhaus.org/query/domain/%s\" target=\"_blank\"> [CLIQUE NO LINK de verificação]</a>" >> %s; echo "<p></p>" >> %s' % (dominioInformado, dominioInformado, relatorio, relatorio))
        # Adiciona no arquivo sapiens.html as contramedidas para as ações 1.1.3
        pagina=open("/var/www/html/sapiens.html","a")
        pagina.write("""
        <p><h4>Ferramentas utilizadas: lista pública de spam.</h4></p><p><h4>
        <p><h4>Contramedidas: nao há</h4></p>
        """)
        pagina.close()



##############################################################################
##############################################################################
###
################################################
###     1.1.8 Examine target using Search Engines
################################################
###                                      ##### PASSO 04 ####
         ################################################
##############################################################################
##############################################################################
# As buscas em Search Engines serão realizada com o comando THEHARVESTER, programa default na distribuição KALI
def A_1_1_8():
        # Limpa a tela do usuário
        comando('clear')
        # Executa o banner explicando o passo 04
        banner_A_1_1_8()
        # Insere linhas com "#" no "relatorio"
        comando('echo " PASSO 04 - Busca por informações indexadas nos motores de busca (A 1.1.8 do ISSAF) <br />" >> %s' % relatorio)
```

```
        comando('echo
"##############################################################################
############################################################<br />" >> %s' %
relatorio)
        comando('echo "<p></p><p><h3>DADOS INDEXADOS PELO GOOGLE (E-
mail)</h3></p>" >> %s' % relatorio)
        # Utiliza a ferramenta "theharvester" para encontrar dados indexados pelo motor de
busca "Google"
        comando('theharvester -d %s -b google -v -c -n -t > %s' % (dominioInformado,
intermediario))
        # Cria o arquivo "listaEmails" com emails do domínio alvo encontrados pela
ferramenta "theharvester" no google
        comando('cat %s | egrep "@$d" >> %s/listaEmails' % (intermediario, diretorioUso))
        # Cria o arquivo "hostsEncontrados" com IP relacionados ao domínio alvo
encontrados pela ferramenta "theharvester" no google. Esse arquivo
        # será usado na função "B_1_1_28"
        comando('cat %s | egrep \'[0-9]:[A-Za-z]\' | sed \'s/:/ -> /g\' >> %s/hostsEncontrados'
% (intermediario, diretorioUso))
        # Insere no arquivo "sapiens.html" todos os emails com o domínio do alvo
encontrados no google
        comando('cat %s/listaEmails | sed \'s-$-<p></p>-g\' | sed 1d >> %s' % (diretorioUso,
relatorio))
        # Insere no arquivo "sapiens.html" todos os hosts relacionados ao domínio alvo que
foram encontrados no google
        comando('echo "<p></p><p><h3>DADOS INDEXADOS PELO GOOGLE
(hosts)</h3></p>" >> %s' % relatorio)
        comando('cat %s/hostsEncontrados | sed \'s-$-<p></p>-g\' >>  %s' %
(diretorioUso,relatorio))

        # Adiciona no arquivo sapiens.html as contramedidas para as ações 1.1.8
        pagina=open('/var/www/html/sapiens.html','a')
        pagina.write("""
        <p><h4>Ferramenta utilizada: THEHARVESTER</h4></p>
        <p><h4>Contramedidas:</h4></p>
        <p><h4>1. Inclua no arquivo robots.txt páginas, que contenham informações
sensíveis (essas informações continuarão disponíveis para usuários do site, mas não
serão indexadas pela maioria dos motores de busca, ou seja, não serão apresentadas em suas
buscas;</h4></p>
        <p><h4>2. Remova dados sensíveis/confidenciais encontrados pelos motores de
busca. Alternativamente, coloque esses dados "atrás" de um mecanismo de autenticação
apropriado.</h4></p>
        <p><h4>Link sobre operadores do google:<a
href="http://www.google.com/help/operators.html" target="_blank"> --> [CLIQUE NO
LINK]</a></h4></p>
        <p><h4>Link sobre robots.txt:<a href="http://www.robotstxt.org/wc/exclusion.html"
target="_blank"> --> [CLIQUE NO LINK]</a></h4></p>
        """)
```

```
        pagina.close()

##############################################################################
##############################################################
###
################################################
###     1.1.18 Email Systems User Account Enumeration
         ################################################
###                                ##### PASSO 05 ####
         ################################################
##############################################################################
##############################################################
#       função não foi testada em nenhum servidor que funcionou
def A_1_1_18():
        # Limpa a tela do usuário
        comando('clear')
        # Executa o banner explicando o passo 05
        banner_A_1_1_18()
        print " "
        # Cria a variável "respSmtp" que guarda um valor ditigado pelo usuário. Caso o
usuário digite "sim" será realizada uma tentativa
        # de inumeração de usuários SMTP, se for digitado "não" o programa executará o
passo seguinte.
        respSmtp = raw_input('\n\nEnumeração de contas de usuário é uma ação considerada
ativa e pode expor seu IP.\nDeseja enumerar possíveis conta de usuários no servidor de
email do domínio alvo?\n\n>>> SIM\n>>> NÃO\n\n>>> ')
        # Cria um loop que, enquanto o usuaŕio não digitar "sim" ou "não" o programa não
prossegue.
        while respSmtp != "sim" and respSmtp != "SIM" and respSmtp != "nao" and
respSmtp != "não" and respSmtp != "NAO" and respSmtp != "NÃO":
                respSmtp = raw_input('\nDigite apenas SIM ou NÃO.\n\n>>> SIM\n>>>
NÃO\n\n>>> ')
        if respSmtp == "sim" or respSmtp == "SIM" or respSmtp == "Sim":
                # Limpa a tela do usuário
                comando('clear')
                # Executa o banner explicando o passo 05
                banner_A_1_1_18()
                # Insere linhas com "#" no "relatorio"
                comando('echo " PASSO 05  -  Sistema de email: identificação de contas de
usuários<br (A 1.1.18 do ISSAF) />" >> %s' % relatorio)
                comando('echo
"##############################################################################
############################################################<br />" >> %s' %
relatorio)
                comando('echo "<p></p><p><h3>USUÁRIO ENCONTRADOS
EXPLORANDO SERVIDOR SMTP</h3></p>" >> %s' % relatorio)
```

```
		# Cria o arquivo "possiveisUserSmtp" para serem testados usuários SMTP, caso queira inserir novos usuários para teste altere a linha abaixo
		comando('echo "admin\\nroot\\nadministrador\\nadmin123\\nteste\\njoao\\nadministrator\\ntoor" > %s/possiveisUserSmtp; a=`cat %s/enderecoMx | sed -n \'1p\'`; smtp-user-enum -vv -M RCPT -U %s/possiveisUserSmtp -t $a > %s/saidaSmtpUser 2> /dev/null' % (diretorioUso, diretorioUso, diretorioUso, diretorioUso))
		# Adiciona no arquivo sapiens.html as contramedidas para as ações 1.1.18
		pagina=open("/var/www/html/sapiens.html","a")
		pagina.write("""
		<p><h4>Ferramentas: SMTP-USER-ENUM</h4></p>
		<p><h4>Contramedidas:</h4></p>
		<p><h4>1.Algumas organizações adicionam um elemente randômico nos endereços de email para dificultar ataques de spamming e phishing e facilitar sua detecção. Mesmo essa medida funcionando em certo grau, torna os endereços menos "estéticos", sendo por isso evitado em algumas organizações.</h4></p>
		<p><h4>2.Implementar políticas para restringir a descoberta e uso de endereços de email (ex: alguma companhias não incluem mais os endereços de email nos cartões pessoais, ao contrário, orientam os empregados a anotar apenas quando expressamente requerido para comunicação de negócios.</h4></p>
		<p><h4>3.Implemente Sistemas de Detecção de Intrusão (IDS) para detectar atividades de coleta de emails nos servidores de email. "mail to:")</h4></p>
		<p><h4>4.Implemente filtros para frustrar levantamento de informações (ex: desabilite comandos SMTP desnecessários; desabilite a fução de relay do email; utilize timeout nas conexões; etc)</h4></p>
		""")
		pagina.close()

##############################################################################
##############################################################################
###
##############################################
###     1.1.23 DNS Interrogation - Perform Zone Transfer on Primary, Secondary and ISP name server  ##############################################
###                                 ### PASSO 06 ###
##############################################
##############################################################################
##############################################################################
def A_1_1_23():
	# Limpa a tela do usuário
	comando('clear')
	# Executa o banner explicando o passo 06
	banner_A_1_1_23()
	print " "
```

```
	# Cria a variável "respTran" que guarda um valor ditigado pelo usuário. Caso o usuário digite "sim" será realizada uma tentativa
	# de transferência de zona (utilizando o comando 'host' e usando os endereços dos servidores DNS presentes no arquivo 'nomeDns', se for	# digitado "não" o programa executará o passo seguinte.
	respTran = raw_input('Deseja realizar uma tentativa de transferência de zona contra o(s) servidor(es) DNS do domínio alvo?\nEssa técnica pode garantir informações valiosas como o nome e IP de todas as máquinas do domínio alvo,\nmas é uma ação "ruidosa" e, sem a devida autorização, pode ser considerada ilegal.\n\n>>> SIM\n>>> NÃO\n\n>>> ')
	# Cria um loop que, enquanto o usuario não digitar "sim" ou "não" o programa não prossegue.
	while respTran != "sim" and respTran != "SIM" and respTran != "nao" and respTran != "não" and respTran != "NAO" and respTran != "NÃO":
		respTran = raw_input('\nDigite apenas SIM ou NÃO.\n\n>>> SIM\n>>> NÃO\n\n>>> ')
	if respTran == "sim" or respTran == "SIM" or respTran == "Sim":
		# Limpa a tela do usuário
		comando('clear')
		# Executa o banner explicando o passo 05
		banner_A_1_1_23()
		# Insere linhas com "#" no "relatorio"
		comando('echo "PASSO 06 -Verificação se os servidores DNS estão vulneráveis a Transf de Zona (A 1.1.23 do ISSAF). <br />" >> %s' % relatorio)
		comando('echo "##########################################################################################################<br />" >> %s' % relatorio)
		# Insere texto no arquivo "sapiens.html"
		comando('echo "<p></p><p><h3>TENTATIVA DE TRANSFERÊNCIA DE ZONA</h3></p>" >> %s' % relatorio)
		# Realiza a tentativa de transferência de zona contra o primeiro DNS server do arquivo "nomeDns" e redireciona a saída para o arquivo
		# "interZona"
		comando('a=`sed -n 1p %s/nomeDns`; host -l %s $a 2> /dev/null > %s/interZona' % (diretorioUso, dominioInformado, diretorioUso))
		# Realiza a tentativa de transferência de zona contra o segundo DNS server do arquivo "nomeDns" e redireciona a saída para o arquivo
		# "interZona"
		comando('a=`sed -n 2p %s/nomeDns`; host -l %s $a 2> /dev/null >> %s/interZona' % (diretorioUso, dominioInformado, diretorioUso))
		# Filtra do arquivo "interZona" os IP e edereços dos hosts encontrados gerados pela transferência de zona.
		comando('cat %s/interZona | sort | uniq | sed \'s/has address/-->/g\' | egrep \'^[a-z]\' | egrep \'has address\' | sed \'s-$-<p></p>-g\' >> %s' % (diretorioUso, relatorio))
		# Adiciona no arquivo sapiens.html as contramedidas para as ações 1.1.23
		pagina=open("/var/www/html/pentest/sapiens.html","a")
		pagina.write("""
```

<h4>Ferramentas: HOST</h4>

<h4>Contramedidas:</h4>

<h4>1. Separar servidores DNS internos e externos (Split-DNS - A contiguração Split DNS consiste em um servidor interno com a a base de dados de todos os nomes DNS da organização e o servidor externo que sabe apenas como resolver nomes relacionados a presença externa da organização, como redirecionadores de email e servidores web. Isso previne que informações internas sejam acessadas pelo mundo externo; </h4>

<h4>2. Em servidores Windows 2000 utilize active directory integrado com servidores DNS internos, já servidores DNS externos devem ficar segregados do domínio do Windows. Não utilize o servidor DNS externo como redirecionador para o servidor DNS interno.

</h4>

<h4>3.Restrinja a transferência de zona para uma lista específica de servidores confiáveis. Configure servidores DNS primários para realizar transferência de zona apenas para seu servidor secundário ou servidores escravos. Caso a transferência mova todos os records para uma zona específica de um servidor para outro, é extremamente importante não transferir o forward lookup zone de um servidor que contenha informações de domínio para qualquer servidor fora do domínio.</h4>

<h4>4. Bloqueie a porta 53/TCP nos firewalls de borda. AXFR trabalha sobre o protocolo TCP, enquanto resoluções de nome normais utilizam 53/UDP;</h4>

<h4>5. Desabilitar updates dinâmicos nos servidores DNS. As versões mais recentes de servidores DNS possuem opção para update dinâmico da zona, integrando com serviços de rede como WINS e DHCP. Esse recurso deve ser desabilitado para servidores DNS externos. </h4>

<h4>6. Não configure HINFO records. Host Information Record (HINFO) é extremamente informativo e pouco funcional. Isso é usado para declarar o tipo de computador e sistema operacional de um host. Essa informação pode ser usada para ações de fingerprint na rede e isso não é recomendado.</h4>

<h4>7. Rode o DNS com usuário sem privilégio. Servidores de nome são sucetíveis a comprometimentos de root através de ataques buffer overflow, quando DNS daemon está sendo executado como root. É mais seguro executar o DNS como usuário comum para minimizar danos caso o servidor seja compromentido. </h4>

<h4>8. Execute o DNS em uma jaula chroot. O dano em um ataque bem sucedido pode ser limitado caso o servidor esteja em um ambiente chroot-ed. No sitema Unix o chroot.</h4>

<h4>9. Mantenha em segurança o arquivo system/registry. É recomendado manter os arquivos relevantes do servidor DNS seguros com configuraçẽos de propriedade e permissão. Para ambientes Microsoft Windows as entradas de registros também necessitam ser protegidas. </h4>

<h4>10. Desabilite todos os serviços desnecessários nos servidores DNS. DNS devem ser configurados com o mínimo de serviços e aplicações rodando para reduzir as chances de comprometimento devido a aplicações vulneráveis.</h4>

```
                <p><h4>11. Atualize os servidores com as correções de segurança mais
atuais. Servidores DNS devem ser regularmente "patched" com as
        correões e atualizações para vulnerabilidades conhecidas.</h4></p>
                <p><h4>12. Habilite logins para transações. Configure um sistema de
autenticação e log em uma base regular. Análise de logs identificará
atividades maliciosas e concederá um alarme prematuro em caso de tentativas de
ataque.</h4></p>
                """)
                pagina.close()

        ##########################################################################
##########################################################################
##
###
############################################
###     1.1.24 DNS Interrogation - Perform Zone Transfer by dictionary attack
           ############################################
###                             ### PASSO 07 ###
############################################
##########################################################################
##########################################################################
def A_1_1_24():
        # Limpa a tela do usuário
        comando('clear')
        # Executa o banner explicando o passo 07
        banner_A_1_1_24()
        print " "
        # Cria a variável "respInter" que guarda um valor ditigado pelo usuário. Caso o
usuário digite "sim" serão realizadas consultas ao
        # servidor(es) DNS do domínio alvo utilizando um dicionário contido no arquivo
"dicionario1.txt". Caso o usuário digite "não" será executado
        # o passo seguinte.
        respInter = raw_input('Deseja realizar um "ataque" de dicionário contra o serviodr
DNS??\nEssa técnica pode garantir informações valiosas como o nome e IP de muitas
máquinas do domínio alvo, \nmas é uma ação "ruidosa".\n\n>>> SIM\n>>> NÃO\n\n>>> ')
        # Cria um loop que, enquanto o usuaŕio não digitar "sim" ou "não" o programa não
prossegue.
        while respInter != "sim" and respInter != "SIM" and respInter != "nao" and respInter
!= "não" and respInter != "NAO" and respInter != "NÃO":
                respInter = raw_input('\nDigite apenas SIM ou NÃO.\n\n>>> SIM\n>>>
NÃO\n\n>>> ')
        if respInter == "sim" or respInter == "SIM" or respInter == "Sim":
                # Limpa a tela do usuário
                comando('clear')
                # Executa o banner explicando o passo 06
                banner_A_1_1_24()
                # Insere linhas com "#" no "relatorio"
```

```
		comando('echo "PASSO 07 - Utilização de dicionário para encontrar host listados em um servidor DNS (A 1.1.24 do ISSAF) .<br />" >> %s' % relatorio)
		comando('echo "##########################################################################################################<br />" >> %s' % relatorio)
		# Criando o arquivo "dicionario1.txt", utilizado para fazer "ataque" de dicionário no servidor DNS do domínio alvo
		comando('echo "access1\\nacervo\\nadmin\\nagenciadenoticias\\nav1\\nbrs\\ncaptiveportal\\ncorreio\\ncorreio2\\nD20\\ndialin\\ncampusonline1\\ncampusonline2\\nwww\\ncampusonline3\\nhmlu\\negressos\\nforbio\\nfsw\\nmantis\\npaf\\ntestlink\\ngenforce\\nhera\\nisg\\nhmlu\\nlyncdiscover\\nlyncweb\\nmail\\nmail-out\\nmail1\\nmeet\\nmemoria\\nmidia\\nmoodle\\nnlb\\nopenppm\\nponto\\npontoweb\\nadmin\\nhmlu\\nposti\\npublicacoes\\nwww\\nrepositorio\\nS10\\nmatricula\\nmatricular\\nsip\\nsoac\\nsoac-hmlu\\nwww\\nwebapps\\nwebcon1\\nwifi\\ncpd\\ndiretoria\\nfone\\nftp\\nintranet\\nmail\\nmailinterno\\nmx1\\nmx2\\nns1\\nns2\\ns3\\nopenbravo\\npop\\npop3\\nposto\\nsmaxx\\nsmtp\\nvnc\\nvpn\\nvpn1\\nvpn2\\nvpn4\\nwebmail" > %s/dicionario1.txt' % diretorioUso)
		# Realiza consulta DNS utilizando nomes contidos no arquivo "dicionairo1.txt". Os nomes que existirem no servidor DNS serão
		# salvos no arquivo "sapiens.html"
		comando('echo "<p></p><p><h3>ATAQUE DE DICIONÁRIO CONTRA DNS</h3></p>" >> %s' % relatorio)
		# Utilizando um laço for, de forma que cada linha do arquivo "dicionario1.txt" seja utilizada com o comando "host". A saída desse
		# comando é redirecionada para o arquivo "subdominio.txt". Em seguida, esse arquivo gerado é filtrado para que permaneçam apenas
		# as linhas com subdomínios encontrados (has address) e essa saída é redirecionada para o arquivo "relatorio"
		comando('for ip in $(cat %s/dicionario1.txt);do host $ip.%s;done | grep \'has add\' >> %s/subdominios.txt; cat %s/subdominios.txt | sort | uniq | sed \'s/has address/-->/g\' | sed \'s-$-<p></p>-g\' >> %s' % (diretorioUso, dominioInformado, diretorioUso, diretorioUso, relatorio))
		# Adiciona no arquivo sapiens.html as contramedidas para as ações 1.1.24
		pagina=open('/var/www/html/sapiens.html','a')
		pagina.write("""
		<p><h4>Ferramenta: HOST </h4></p>
		<p><h4>Contramedidas: </h4></p>
		<p><h4>1.Sempre que possível, evite utilizar nomes comuns (fácil adivinhar) para servidores críticos. Um nome randômico com poucos caraceteres pode ser anexado nos nomes da rede para tornar mais difícil os ataques de dicionário (ex: ftpsd3.alvo.com em detrimento de ftp.alvo.com);</h4></p>
		<p><h4>2. Alguns padrões requerem o estabelecimento de conveções para os nomes e em outros casos não é conveniente algerar o nome por motivos estéticos ou razões práticas (Ex: www-gt4.alvo-dominio.com em detrimento de www.alvodominio.com). Portanto, não tem sentido em alterar o nome de servidores públicos. Considere essa solução
```

```
apenas para servidores públicos que proveem serviços para um número restrito de usuários e organizações (ex: intranet para acesso remoto dos usuários de uma empresa);</h4></p>
                <p><h4>3. Estabelecer protocolos de autenticação DNS, se possível. Restringir transferência de zona apenas para servidores autorizados;</h4></p>
                <p><h4>4. Permitir transferência de zona somente com a diretiva allow-transfer no named.conf;</h4></p>
                <p><h4>5. Não permitir qualquer conexão "inbond" não autorizada para porta 53/TCP;</h4></p>
                <p><h4>6. Implementar opção "notify" em servidores DNS Microsoft;</h4></p>

                <p><h4>Dicas: • http://www.ietf.org/rfc/2845.txt • http://www.ietf.org/rfc/2930.txt • http://www.ietf.org/rfc/3008.txt • http://www.ietf.org/internet-drafts/draft-ietf-dnsext-dnssec-roadmap-06.txt</h4></p>
                """)
                pagina.close()

################################################################################
################################################################################
###
##############################################
###     1.1.26 Mirror Target Web Site
          ##############################################
###                     ### PASSO 08 ###
##############################################
################################################################################
################################################################################
def A_1_1_26():
        # Limpa a tela do usuário
        comando('clear')
        # Executa o banner explicando o passo 08
        banner_A_1_1_26()
        print " "
        # Cria a variável "respEspe" que guarda um valor ditigado pelo usuário. Caso o usuário digite "sim" o endereço web do domínio alvo
        # será clonado para avaliação de vulnerabilidades web. Caso o usuário digite "não" será executado o passo seguinte.
        respEspe = raw_input('\n\nDeseja realizar uma cópia do site alvo?\nEssa técnica pode demorar alguns minutos e caracteriza-se por ser muito ruidosa.\n\n>>> SIM\n>>> NÃO\n\n>>> ')
        # Cria um loop que, enquanto o usuaŕio não digitar "sim" ou "não" o programa não prossegue.
        while respEspe != "sim" and respEspe != "SIM" and respEspe != "nao" and respEspe != "não" and respEspe != "NAO" and respEspe != "NÃO":
                respEspe = raw_input('\nDigite apenas SIM ou NÃO.\n\n>>> SIM\n>>> NÃO\n\n>>> ')
        if respEspe == "sim" or respEspe == "SIM" or respEspe == "Sim":
```

```
                # Limpa a tela do usuário
                comando('clear')
                # Executa o banner explicando o passo 08
                banner_A_1_1_26()
                # Insere linhas com "#" no "relatorio"
                comando('echo "PASSO 08 -Cópia do site alvo (A 1.1.26 do ISSAF)<br />"
>> %s' % relatorio)
                comando('echo
"##############################################################################
############################################################<br />" >> %s' %
relatorio)
                comando('echo "<p></p><p><h3>CÓPIA DO SITE</h3></p>">> %s' %
relatorio)
                # Muda o diretório atual para /pentest, com o intuito do programa "httrack"
ser executado nesse diretório
                comando('cd %s/' % diretorioUso)
                # Executa o programa "httrack" contra o domínio alvo. Qualquer saída
legítima ou de erro serão redirecionadas para o arquivo "lixo"                # e
/dev/null, respectivamente. Em seguida, alguns arquivos desnecessários são deletados. A
cópia do site ficará arquivada no                # diretório atual (/pentest).
                comando('httrack -Q -q www.%s 2> /dev/null 1> lixo; rm -r hts* index.html
backblue.gif fade.gif lixo' % dominioInformado)
                # Copia todos o site para o diretório /pentest e deleta qualquer arquivo
deixado no diretório atual
                comando('cp -r www.%s/ %s/www.%s; rm -r www.%s/ 2> /dev/null' %
(dominioInformado, diretorioUso, dominioInformado, dominioInformado))
                # Insere no arquivo "sapiens.html" o link para acessar o diretório onde o site
está arquivado
                comando('echo "O site foi clonado e encontra-se no seguinte diretório: <a
href=\"pentest/www.%s\" target=\"_blank\"> [CLIQUE NO LINK]-> www.%s
</a><p></p>" >> %s' % (dominioInformado, dominioInformado, relatorio))
                # Adiciona no arquivo sapiens.html as contramedidas para as ações 1.1.26
                pagina=open('/var/www/html/sapiens.html','a')
                pagina.write("""
                <p><h4>Ferramentas: HTTRACK</h4></p>
                <p><h4>Contramedidas:</h4></p>
                <p><h4>1. Para evitar o vazamento de informações, organizações devem
certificar-se que: comentários nas páginas web em produção não                contém
dados sensíveis; </h4></p>
                <p><h4>2. Informações confidenciais devem ser separadas em repositórios
diferentes das informações públicas. O acesso a essas
                informações devem ser restritos e controlados; </h4></p>
                <p><h4>3. Implementar um correto controle de sessão, evitando acesso a
informações restritas por usuários não autorizados ou por                usuários que não
iniciaram a sessão corretamente. </h4></p>
                """)
                pagina.close()
```

```
#       B.2 NETWORK MAPPING (SCANNING, OS FINGERPRINTING AND
ENUMERATION)

##############################################################################
##############################################################################
###
##########################################
###     1.1.28 Identify Live Hosts
##########################################
###                     ### PASSO 09 ###
##########################################
##############################################################################
##############################################################################

def B_1_1_28():
        # Limpa a tela do usuário
        comando('clear')
        # Executa o banner explicando o passo 09
        banner_B_1_1_28()
        # cria o arquivo hostsAtivos.txt com todos os IP que estão ativos na rede alvo, para
tanto, utiliza os arquivos intermediários criados nos passos anteriores e redireciona a saída
para o arquivo "range.txt".
        comando('ping -c 1 %s | cut -s -d "(" -f2 | cut -d ")" -f1 | head -1 >> %s/range.txt 2>
/dev/null; cat %s/ipDns %s/enderecoMx >> %s/range.txt; cat %s/subdominios.txt 2>
/dev/null | egrep -o \'([0-9]{1,3}\.){3}[0-9]{1,3}\' >> %s/range.txt; cat %s/range.txt | uniq >
%s/todosIp' % (dominioInformado, diretorioUso, diretorioUso, diretorioUso, diretorioUso,
diretorioUso, diretorioUso, diretorioUso, diretorioUso))
        # Insere linhas com "#" no "relatorio"
        comando('echo "PASSO 09 - Identificação de hosts ativos (B 1.1.28 do ISSAF). <br
/>" >> %s' % relatorio)
        comando('echo
"##############################################################################
##########################################################<br />" >> %s' %
relatorio)
        comando('echo "<p></p><p><h3>HOSTS ATIVOS NO RANGE ALVO</h3></p>"
>> %s' % relatorio)
        # Utiliza, com o comando NMAP (-PE - manda um ICMP type 8; -sn - sem realizar
port scan; -n - sem realizar resolução de nomes), um port scan
        # no arquivo "range.txt" e direciona a saída para o arquivo "hostsAtivos.txt"
        comando('nmap -PE -n -sn -iL %s/range.txt -oG %s/hostsAtivos.txt 1> /dev/null' %
(diretorioUso, diretorioUso))
        # Filtra dos arquivos "subdominio.txt" e "hostsEncontrados" apenas os endereços IP e
redireciona a saída para o arquivo "hostsAtivos.txt"
```

```
        comando('cat %s/subdominios.txt %s/hostsEncontrados 2> /dev/null | egrep -o \'([0-9]{1,3}\.){3}[0-9]{1,3}\' | sort | uniq >> %s/hostsAtivos.txt' % (diretorioUso, diretorioUso, diretorioUso))
        # Filtra do arquivo "hostsAtivos.txt" apenas os IP e redireciona para "hostsAtivosSoIP.txt"
        comando('egrep -o \'([0-9]{1,3}\.){3}[0-9]{1,3}\' %s/hostsAtivos.txt | sort | uniq >> %s/hostsAtivosSoIp.txt' % (diretorioUso,diretorioUso))
        # Cria no arquivo "relatorio" um link para o arquivo "hostsAtivosSoIp.txt"
        comando('echo "Lista com IP ativos no range obtido com o comando whois e através de consultas DNS: <a href=\"pentest/hostsAtivosSoIp.txt\" target=\"_blank\"> [CLIQUE NO LINK] --> Lista </a><p></p>" >> %s' % relatorio)


##############################################################################
##############################################################################
###
##########################################
###     1.1.29 (B.2.1.1, B.2.1.1.2 e B.2.1.1.3) TCP e UDP port scaning e Banner Grabing
                ##########################################
###                     ### PASSO 10 ###
##########################################
##############################################################################
##############################################################################


def B_1_1_29_B_2_1_1_e_B_2_1_1_2():
        global alvoScan
        # Limpa a tela do usuário
        comando('clear')
        # Executa o banner explicando o passo 10
        banner_B_1_1_29_B_2_1_1_e_B_2_1_1_2()
        print " "
        # Cria a variável "respScan" que guarda um valor ditigado pelo usuário. Caso o usuário digite "sim" será realizado um scaneamento
        # das portas TCP e UDP do host alvo. Caso o usuário digite "não" será executado o passo seguinte.
        respScan = raw_input('Deseja realizar um scaneamento dos serviços TCP e UDP em algum dos IP ativos no range alvo?\nPara encontrar alguma vulnerabilidade, sugere-se que seja respondido "SIM".\n\n>>> SIM\n>>> NÃO\n\n>>> ')
        # Cria um loop que, enquanto o usuário não digitar "sim" ou "não" o programa não prossegue.
        while respScan != "sim" and respScan != "SIM" and respScan != "nao" and respScan != "não" and respScan != "NAO" and respScan != "NÃO":
                respScan = raw_input('\nDigite apenas SIM ou NÃO.\n\n>>> SIM\n>>> NÃO\n\n>>> ')
        if respScan == "sim" or respScan == "SIM" or respScan == "Sim":
                #limpa a tela
```

```
                comando('clear')
                comando('echo "Os seguintes IP estão ativos no range alvo\n"')
                # Apresenta na tela a lista de hosts ativos na rede alvo.
                comando('cat %s/hostsAtivosSoIp.txt' % diretorioUso)
                # Cria a variável "alvoScan", que determinará qual ip será executado o
scaneamento.
                alvoScan = raw_input('\n\nQual IP deve ser scaneado?\n\n>>> ')
                # Verifica se o valor inserido pelo usuário está no formato de um endereço IP.
                formatoIp = commands.getoutput('if echo "%s" | egrep -o \'([0-
9]{1,3}\.){3}[0-9]{1,3}\'; then echo "HeIp";else echo "naoHeIp";fi' % alvoScan)
                while formatoIp == "naoHeIp":
                        alvoScan = raw_input('\nO valor inserido não está no formato de um
endereço IP. Digite o endereço no seguinte formato:\nEx: 10.11.22.33\n\n>>> ')
                        formatoIp = commands.getoutput('if echo "%s" | egrep -o \'([0-
9]{1,3}\.){3}[0-9]{1,3}\'; then echo "HeIp";else echo "naoHeIp";fi' % alvoScan)
                # Limpa a tela do usuário
                comando('clear')
                # Executa o banner explicando o passo 10
                banner_B_1_1_29_B_2_1_1_e_B_2_1_1_2()
                # Insere linhas com "#" no "relatorio"
                comando('echo "PASSO 10 - Scaneamento de serviços TCP e UDP (1.1.29 -
B.2.1.1 e B.2.1.1.2 do ISSAF).<br />" >> %s' % relatorio)
                comando('echo
"#########################################################################
#########################################################<br />" >> %s' %
relatorio)
                comando('echo "<p></p><p><h3>SCANEAMENTO - SERVIÇOS TCP
ATIVOS</h3></p>" >> %s' % relatorio)
                # B.2.1.1.1 Utiliza o comando "nmap" para realizar um scaneamento e banner
grabing nas portas TCP default e redireciona a saída para                # o arquivo
"dominio_portasTcp.txt". Em seguida, é gerada um link no arquivo "relatorio" para acessar o
novo arquivo criado.
                comando('nmap -A %s >> %s/%s_PortasTcp.txt; echo "Arquivos com lista
dos serviços TCP encontrados: <a href=\"pentest/%s_PortasTcp.txt\" target=\"_blank\">
[CLIQUE NO LINK]--> link</a><p></p>" >> %s' % (alvoScan, diretorioUso, alvoScan,
alvoScan, relatorio))
                # B.2.1.1.2 Utiliza o comando "nmap" para realizar um scaneamento e banner
grabing nas portas principais portas UDP e redireciona a                # saída para o
arquivo "dominio_portasUdp.txt". Em seguida, é gerada um link no arquivo "relatorio" para
acessar o novo arquivo                # criado.
                comando('echo "<p></p><p><h3>SCANEAMENTO - SERVIÇOS UDP
ATIVOS</h3></p>" >> %s' % relatorio)
                comando('nmap -sU -p 22,23,25,53,67,68,123,161,162,546,547,1723 %s >>
%s/%s_PortasUdp.txt; echo "Arquivos com lista dos serviços UDP encontrados:<a
href=\"pentest/%s_PortasUdp.txt\" target=\"_blank\"> [CLIQUE NO LINK] -->
link</a><p></p>" >> %s' % (alvoScan, diretorioUso, alvoScan, alvoScan, relatorio))
```

```
                # Adiciona no arquivo sapiens.html as contramedidas para as ações 1.1.26
                pagina=open('/var/www/html/sapiens.html','a')
                pagina.write("""
                <p><h4> Contramedidas:</h4></p>
                <p><h4> 1. Implementar firewalls, permitindo apenas o que for
absolutamente necessário; </h4></p>
                <p><h4> 2. Restringir endereços de origem nos firewalls caso algum serviço
não deva ser acessado por ninguém (ex: servidores de administração restritos
devem ser acessados apenas pelos hosts dos administradores) </h4></p>
                <p><h4> 3. alterar a versão e nome do produto nos banners dos serviços, isso
dificultará a identificação correta dos sistemas por
                parte de um atacante. Contudo, essa solução não é a prova de balas e não
torna o sistema mais seguro.</h4></p>
                """)
                pagina.close
                # Executa a função "B_2_1_10"
#               B_2_1_10()
                # Executa a funçao "B_encontrar_cve()"
                B_encontrar_cve()
                # Executa a funçao "integraNessus()"
                integraNessus()



################################################################################
################################################################################
###
#############################################
###     1.1.29 (B.2.1.10) SYSTEMS ENUMERATIONS
                  #############################################
###                     ### PASSO 11 ###
#############################################
################################################################################
################################################################################

def B_2_1_10():
        # Limpa a tela do usuário
        comando('clear')
        # Executa o banner explicando o passo 11
        banner_B_2_1_10()
        print " "
        # Cria a variável "respEnum" que guarda um valor ditigado pelo usuário. Caso o
usuário digite "sim" serão realizadas ações de enumeração
        # através do comando "nmap". Caso o usuário digite "não" será executado o passo
seguinte.
        respEnum = raw_input('\n\nDeseja realizar técnicas de enumeração contra o IP
informado?\nEssa ação pode trazer mais detalhes sobre cada\n um dos serviços UDP e TCP
```

```
encontrados, mas levará alguns minutos para ser concluída\n\n>>> SIM\n>>> NÃO\n\n>>>
')
        # Cria um loop que, enquanto o usuário não digitar "sim" ou "não" o programa não
prossegue.
        while respEnum != "sim" and respEnum != "SIM" and respEnum != "nao" and
respEnum != "não" and respEnum != "NAO" and respEnum != "NÃO":
                respEnum = raw_input('\nDigite apenas SIM ou NÃO.\n\n>>> SIM\n>>>
NÃO\n\n>>> ')
        if respEnum == "sim" or respEnum == "SIM" or respEnum == "Sim":
                global alvoScan
                # Limpa a tela do usuário
                comando('clear')
                # Executa o banner explicando o passo 10
                banner_B_2_1_10()
                print " "
                # Insere linhas com "#" no "relatorio"
                comando('echo "PASSO 11 - Enumeração dos serviços ativos (1.1.29 -
B.2.1.10 do ISSAF) System Enumeration            ####<br />" >> %s' % relatorio)
                comando('echo
"################################################################################
############################################################<br />" >> %s' %
relatorio)
                comando('echo "<p></p><p><h3>ENUMERAÇÃO</h3></p>" >> %s' %
relatorio)
                # Cria o diretório "enumeracao", que será o destino dos arquivos gerados
durante o passo de enumeração
                comando('mkdir %s/enumeracao' % diretorioUso)
                # Realiza ações de enumeração utilizando scripts do NMAP e redireciona a
saída para 3 arquivos distintos no diretório criado
                # anteriormente ("enumeracao").
                comando('nmap --script default %s >> %s/enumeracao/script_default_%s 2>
/dev/null' % (alvoScan, diretorioUso, alvoScan))
                comando('nmap --script discovery %s >> %s/enumeracao/script_dicovery_%s
2> /dev/null' % (alvoScan, diretorioUso, alvoScan))
                comando('nmap --script exploit %s >> %s/enumeracao/script_exploit_%s 2>
/dev/null' % (alvoScan, diretorioUso, alvoScan))
                # Insere no arquivo "relatorio" um link para o usuário acessar o diretório
"pentest" onde os arquivos gerados pelo "nmap" foram
                # salvos.
                comando('echo "Diretório com resultado de enumeração realizada contra
serviços ativos encontrados no IP alvo:<a href=\"pentest/enumeracao\" target=\"_blank\">
[CLIQUE NO LINK]--> link</a><p></p>Foram gerados três arquivos: script_default_%s,
script_dicovery_%s e script_exploit_%s. Cada um deles foi gerado com uma série de
comandos específicos do NMAP, dedicados a obter diferentes informações.<p></p>Análise
os arquivos gerados para encontrar informações úteis." >> %s' % (alvoScan, alvoScan,
alvoScan, relatorio))
                # Adiciona no arquivo sapiens.html as contramedidas para as ações 1.1.26
```

```
                pagina=open('/var/www/html/sapiens.html','a')
                pagina.write("""
                <p><h4> Contramedidas:</h4></p>
                <p><h4> 1. Deixar disponível na internet apenas informações extremamente
necessárias sobre versões de softwares e serviços presentes em sua
rede; </h4></p>
                """)
                pagina.close


def B_encontrar_cve():
        # Importa o valor da variável criada na função "B_1_1_29_B_2_1_1_e_B_2_1_1_2"
        global alvoScan
        # Cria a variável "dirVfeed" cujo valr será o diretório onde o programa "vfeed" foi
instalado.
        dirVfeed = commands.getoutput('locate "/vFeed/" | egrep -o \'.*/vFeed\' | sed -n 1p')
        # Cria arquivo "inter2" apenas com as versões dos serviços TCP e UDP encontradas,
com base nos arquivos gerados na função #
"B_1_1_29_B_2_1_1_e_B_2_1_1_2()"
        comando('cat %s/%s_PortasTcp.txt | egrep -E \'^[0-9].*/tcp\' | egrep -o [A-Z].* | sort |
uniq > %s/%s_inter2' % (diretorioUso, alvoScan, diretorioUso, alvoScan))
        comando('cat %s/%s_PortasUdp.txt | egrep -E \'^[0-9].*/tcp\' | egrep -o [A-Z].* | sort |
uniq >> %s/%s_inter2' % (diretorioUso, alvoScan, diretorioUso, alvoScan))
        # Cria a variável "respCve" com o valor de "no". Essa variável é reponsável por
manter o "while" em execução. Enquanto "respCve" não for alterado pelo usuário para um
valor igual a "gerar" ou "GERAR" o programa continuará executando o "while".
        respCve = "no"
        # Limpa a tela
        comando('clear')
        print " "
        # Imprime mensagem na tela
        print "Durante o scanning do IP: %s foram encontrados os seguintes serviços/versões:
" % alvoScan
        print " "
        # Apresenta ao usuário o conteúdo do arquivo "inter2", que contém apenas as versões
dos serviços encontrados na máquina alvo
        comando('cat %s/%s_inter2' % (diretorioUso, alvoScan))
        print " "
        # Cria a variável "respCve1" que reberá um valor digitado pelo usuário. Caso seja
digitado "sim" o programa entrará no "IF" abaixo, caso digite não passará para o
passo seguinte.
        respCve1 = raw_input('Deseja saber se algum desses serviços/versões possui
vulnerabilidades conhecidas (CVE)?\n\n>>> SIM\n>>> NÃO\n\n>>> ')
        # Cria um loop que, enquanto o usuário não digitar "sim" ou "não" o programa não
prossegue.
        while respCve1 != "sim" and respCve1 != "SIM" and respCve1 != "nao" and
respCve1 != "não" and respCve1 != "NAO" and respCve1 != "NÃO":
```

```
                respCve1 = raw_input('\nDigite apenas SIM ou NÃO.\n\n>>> SIM\n>>>
NÃO\n\n>>> ')
        if respCve1 == "sim" or respCve1 == "SIM":
##################PEGAR DIRETÓRIO ATUAL diretorioAtual. Preciso fazer isso se o
diretório for alterado após essa função ser executada
                # Insere linhas com "#" no "relatorio"
                comando('echo "PASSO 12 - Vulnerabilidades conhecidas encontradas    <br
/>" >> %s' % relatorio)
                comando('echo
"#################################################################################
#############################################################<br />" >> %s' %
relatorio)
                # Insere uma linha no arquivo sapiens.html
                comando('echo "<p></p><p><h3>CVE ENCONTRADOS DO IP %s
</h3></p>" >> %s' % (alvoScan, relatorio))
                # Altera o valor da variável "respCve" que era "no". O valor atribuído aqui
será alvo do comando "vfeedcli.py".
                respCve = raw_input('\nQual serviço/versão deseja pesquisar?\n\n>>> ')
                # Executa o comando "vfeedcly.py" contra o valor da variável "respCve". A
saída será direcionada para o arquivo "cve.txt". Esse arquivo será a base para a criação de
outros dois o "cveTratado" e o "soCve". Ambos serão integrados com um laço for para
serem inseridos no arquivo relatório.
                comando('cd %s; python vfeedcli.py -s "%s" >> %s/cve_%s.txt 2> /dev/null'
% (dirVfeed, respCve, diretorioUso, alvoScan))
                # Filtra o conteúdo do arquivo "cve.txt", redirecionando o resultado para o
arquivo "cveTratado" somente com as linhas referentes a CVE geradas pelo comando
"vfeedcli.py"
                comando('cat %s/cve_%s.txt | egrep \'>\' | sort | uniq > %s/cveTratado_%s.txt'
% (diretorioUso, alvoScan, diretorioUso, alvoScan))
                # Filtra o conteúdo do arquivo "cve.txt", redirecionando o resultado para o
arquivo "soCve" somente com os números de CVE geradas pelo comando "vfeedcli.py"
                comando('egrep -o \'|-> CVE-[0-9]{4}-[0-9]{4}\' %s/cve_%s.txt | egrep -o
\'CVE-[0-9]{4}-[0-9]{4}\' | sort | uniq >> %s/soCve_%s.txt' % (diretorioUso, alvoScan,
diretorioUso, alvoScan))
                # Insere as informações contidas nos arquivos "cveTratado" e "soCve" no
arquivo "sapiens.html" criando um link externo para cada CVE listado no arquivo
"soCVE_IP"
                comando('for i in $(seq 1 `wc -l %s/soCve_%s.txt | cut -d " " -f 1`); do sed -n
$i\'p\' %s/cveTratado_%s.txt >> %s; echo "<a
href=\"https://web.nvd.nist.gov/view/vuln/detail?vulnId=`sed -n $i\'p\' %s/soCve_%s.txt`\"
target=\"_blank\"><u><b> [CLIQUE NO LINK] -> `sed -n $i\'p\'
%s/soCve_%s.txt`</b></u></a><p></p>" >> %s; done' % (diretorioUso, alvoScan,
diretorioUso, alvoScan, relatorio, diretorioUso, alvoScan, diretorioUso, alvoScan, relatorio))
                # O valor da variável "respCve" foi passado pelo usuário anteriormente e seu
conteúdo é o nome de algum serviço. Por isso, o programa executará o "while" abaixo
enquanto o usuário não digitar a palavra "gerar". Caso a palavra "gerar" seja passada pelo
usuário o "while" será interrompido e o passo seguinte será executado.
```

```
                while respCve != "gerar" or respCve != "GERAR":
                        respCve = raw_input('\nDeseja pesquisar os CVE de outro
serviço/versão? Caso queira digite o nome do serviço/versão, mas caso queira gerar o
relatório com os CVE já descobertos, digite a palavra "GERAR":\n\n>>> ')
                # Se o valor da variável "respCve" for igual a "gerar" o laço é interrompido e
o passo seguinte será executado.       Caso contrário, todas as etapas executadas com o
comando "vfeedcly.py" serão executadas novamente.
                        if respCve == "gerar" or respCve == "GERAR":
                                break
                        comando('cd %s; python vfeedcli.py -s "%s" > %s/cve_%s.txt 2>
/dev/null' % (dirVfeed, respCve, diretorioUso, alvoScan))
                        comando('cat %s/cve_%s.txt | egrep \'>\' | sort | uniq >
%s/cveTratado_%s.txt' % (diretorioUso, alvoScan, diretorioUso, alvoScan))
                        comando('egrep -o \'|-> CVE-[0-9]{4}-[0-9]{4}\' %s/cve_%s.txt |
egrep -o \'CVE-[0-9]{4}-[0-9]{4}\' | sort | uniq > %s/soCve_%s.txt' % (diretorioUso,
alvoScan, diretorioUso, alvoScan))
                        comando('for i in $(seq 1 `wc -l %s/soCve_%s.txt | cut -d " " -f 1`); do
sed -n $i\'p\' %s/cveTratado_%s.txt >> %s; echo "<a
href=\"https://web.nvd.nist.gov/view/vuln/detail?vulnId=`sed -n $i\'p\' %s/soCve_%s.txt`\"
target=\"_blank\"><u><b> -> `sed -n $i\'p\' %s/soCve_%s.txt`</b></u></a><p></p>" >>
%s; done' % (diretorioUso, alvoScan, diretorioUso, alvoScan, relatorio, diretorioUso,
alvoScan, diretorioUso, alvoScan, relatorio))

alvoScan = "10.21.1.92"

# Função para integrar o scanner "Nessus" com a ferramenta "Sapiens"
def integraNessus():
        global ipAlvoNessus
        # Limpa a tela do usuário
        comando('clear')
        # Executa o banner explicando o passo 10
        banner_integraNessus()
        # Cria a variável "respNessus" que guarda um valor ditigado pelo usuário. Caso o
usuário digite "sim" será executado o scan de
        # vulnerabilidade "Nessus" contra o IP alvo. Caso o usuário digite "não" será
executado o passo seguinte.
        respNessus = raw_input('\n\nVocê deseja executar o scan de vulnerabilidade
NESSUS contra o IP informado? \n\n>>> SIM\n>>> NÃO\n\n>>> ')
        # Cria um loop que, enquanto o usuário não digitar "sim" ou "não" o programa não
prossegue.
        while respNessus != "sim" and respNessus != "SIM" and respNessus != "nao" and
respNessus != "não" and respNessus != "NAO" and respNessus != "NÃO":
                respNessus = raw_input('\nDigite apenas SIM ou NÃO.\n\n>>> SIM\n>>>
NÃO\n\n>>> ')
        # Se o usuário digitar "sim" no passo anterior, será executao o "nessus"
        if respNessus == "SIM" or respNessus == "sim" or respNessus == "s" or respNessus
== "S":
```

```
		# Cria a variável "nessusInst" que verifica se o "nessus" está instalado na máquina.
		nessusInst = commands.getoutput('if service nessusd start 2> /dev/null 1> /dev/null; then echo "instalado"; else echo "nao_instalado";fi')
		# Caso o "Nessus" já esteja instaldo na máquina as linhas a seguir serão executadas
		if nessusInst == "instalado":
			print "O Nessus está sendo iniciado....."
			print " "
			print "Em alguns minutos a TELA DE LOGIN do Nessus aparecerá (no browser). Após isso acontecer digite \"INICIAR\" quando for solicitado."
			print " "
			print "Aguarde..."
			print " "
		# Caso o "Nessus" não esteja instalado na máquina as linhas a seguir serão executadas
		else:
			print "O Nessus não está instalado, instale-o antes de prossguir.\n"
			print "1. Baixe o pacote referente ao seu sistema operacional no link: http://www.tenable.com/products/nessus/select-your-operating-system\n"
			print "2. Após baixado, decompacte o pacote com o comando \"dpkg -i xxxx\"\n"
			print "3. Gere um código de ativação no link: https://www.tenable.com/products/nessus-home\n"
			print "4. Utilizando o código recebido no email cadastrado, gere as credenciais de acesso."
		# Programa aguarda 7 segundos até o serviço levantar
		comando('sleep 7')
		# Executa o arquivo iniciaNessus.py, que abre o browser para inicialização do ambiente gráfico do Nessus
		comando('python iniciaNessus.py 1> /dev/null 2> /dev/null')
		# Cria um loop, que executa o nessus apenas após o usuário digitar INICIAR, presumindo que ele tenha fechado o browser aberto no passo anterior. O fechamento do browser por parte do usuário na etapa anterior é indispensável para o correto funcionamento do programa.
		iniciarNessus = "s"
		while iniciarNessus != "INICIAR" and iniciarNessus != "iniciar":
			iniciarNessus = raw_input('Após fechar a janela digite \"INICIAR\":\n\n>>> ')
		# Coleta credenciais do usuários e IP alvo. Esses dados são inseridos no arquivo executaNessus.py, responsável por automatizar o scaneamento do nessus
		nomeLogin = raw_input('\n\nDigite o login cadastrado para acessar o Nessus: ')
		comando('sed -i \'s/username").send_keys(".*")/username").send_keys("%s")/g\' executaNessus.py' % nomeLogin)
```

```
		senhaLogin = raw_input('\n\nDigite a senha cadastrada para acessar o Nessus:
')
		comando('sed -i
\'s/password").send_keys(".*")/password").send_keys("%s")/g\' executaNessus.py' %
senhaLogin)
		ipAlvoNessus = raw_input('\n\nDigite o IP que deve ser scaneado: ')

		comando('sed -i \'s/textarea.editor-
input.required").send_keys(".*")/textarea.editor-input.required").send_keys("%s")/g\'
executaNessus.py' % ipAlvoNessus)
		comando('sed -i \'s/input.editor-input.required").send_keys(".*")/input.editor-
input.required").send_keys("scan_%s")/g\' executaNessus.py' % ipAlvoNessus)
		# Inicia o scanemanto com o nessus, executando o arquivo
"executaNessus.py", que contém o código para automatizar o Nessus.
		comando('python executaNessus.py')

def comandoUsuario():
	# Limpa a tela do usuário
	comando('clear')
	# Executa o banner explicando o passo 10
	print "\n\nTalvez não tenham sido realizadas todas as ações de uma auditoria de
pentest que o senhor gostaria de ter realizado."
	# Cria a variável "respComdUser" que guarda um valor ditigado pelo usuário. Caso o
usuário digite "sim" será executado o comando que o usuário digitar no shell contra o IP ou
domínio alvo.
	respComdUser = raw_input('\n\nDeseja executar algum comando contra o domínio
ou IP alvo que ainda não foi realizado? \n\n>>> SIM\n>>> NÃO\n\n>>> ')
	# Cria um loop que, enquanto o usuário não digitar "sim" ou "não" o programa não
prossegue.
	while respComdUser != "sim" and respComdUser != "SIM" and respComdUser !=
"nao" and respComdUser != "não" and respComdUser != "NAO" and respComdUser !=
"NÃO":
		respComdUser = raw_input('\nDigite apenas SIM ou NÃO.\n\n>>>
SIM\n>>> NÃO\n\n>>> ')
	# Se o usuário digitar "sim" no passo anterior, será executao o "nessus"
	if respComdUser == "SIM" or respComdUser == "sim" or respComdUser == "s" or
respComdUser == "S":
		# Cria a variável "nessusInst" que verifica se o "nessus" está instalado na
máquina.
		ComdUser = raw_input('\n\nDigite o comando desejado: Obs 1: serão aceitos
apenas comandos reconhecidos pelo bash shell;\nObs 2: insira o comando completo; \n\n>>>
')
		# Insere linhas com "#" no "relatorio"
		comando('echo "PASSO 13 - Comando gerado pelo usuário		<br />"
>> %s' % relatorio)
		comando('echo
"#############################################################################
```

```
##########################################################<br />" >> %s' %
relatorio)
                # Executa o comando sugerido pelo usuário e redireciona o conteúdo para o
arquivo
                comando('%s >> %s' % (ComdUser, relatorio))


def banner_A_1_1_2():
        print " "
        print " "
        print " "
        print " "
        print
"#############################################################################
########################"
        print "################################     EXECUTANDO    PASSO 01
##############################"
        print
"#############################################################################
########################"
        print "
        "
        print "   Obtendo informações sobre domínio e blocos IP através dos Registros
Regionais                "
        print "   de Internet (RIR). Cada país gerencia seus domínio e em um nível global
existem os      "
        print "  \"top level domains\", como: .com, .org, net, etc.
        "
        print "   Exemplo de top level domains de países:
        "
        print "
        "
        print "   http://www.nic.uk/ Reino Unido (.uk dominio)
                "
        print "   http://www.nic.ar/ Argentina (.ar dominio)
        "
        print "
        "
        print "   Existem 5 RIR supervisionados pela ICANN, que são responsáveis por
alocar endereços IP,"
        print "   dominios e número de sitemas autônomos:
        "
        print "   • APNIC (http://www.apnic.net/) - Asia-Pacifico
        "
        print "   • ARIN (http://www.arin.net/)- América do Norte
        "
```

```
        print "   • LACNIC (http://www.lacnic.net/) - América Latina e Caribe
                "
        print "   • RIPE (http://www.ripe.net/) - Europa e Asia central
                "
        print "   • AFRINIC (http://www.afrinic.net/) – Africa
                "
        print "
        "
        print "   EXECUTANDO.... (AGUARDE)
                "
        print "
        "
        print
"##############################################################################
#########################"
        print
"##############################################################################
#########################"
        print
"##############################################################################
#########################"

def banner_A_1_1_3():
        print " "
        print " "
        print " "
        print " "
        print
"##############################################################################
#########################"
        print "#############################      EXECUTANDO    PASSO 02
##################################"
        print
"##############################################################################
#########################"
        print "
        "
        print "   Obtendo informações através de consulta a servidores DNS.
                "
        print "   O servidor de nome(s) autoritativo de um domínio guarda sua zona. A zona é
a lista de        "
        print "   registros do domínio. Embora um único registro DNS seja considerado uma
informação     "
        print "   pública, não é aconselhável permitir acesso a todos os registros de uma zona
de uma"
        print "   só vez (algo conhecido como Transferência de Zona ou AXFR). As
informações contidas na "
```

```
        print "   zona são importantes para \"pentesters\", que tentam realizar o ataque conhecido como         "
        print "   transferência de zona. Os servidores de nome autoritativos podem ser encontrados de duas"
        print "   maneiras. A primeira é através do comando whois, o segundo através do uso da infraestru-"
        print "   tura do DNS. Uma vez encontrado o sevidor de nomes autoritativo, uma requisição do tipo"
        print "   AXFR (Zone Transfer) pode ser tentada; se bem sucedida, a requisição pode garantir infor-"
        print "   mações completas da zona de um determinado domínio.
                "
        print "   Outra informação importante a ser obtida é a versão do serviço DNS em uso. Versões antigas"
        print "   geralmente possuem vulnerabilidades que permitem seu comprometimento por atacantes.  "
        print "
        "
        print "   EXECUTANDO.... (AGUARDE)
                "
        print "
        "
        print "##############################################################################################################"
        print "##############################################################################################################"
        print "##############################################################################################################"


def banner_A_1_1_5():
        print " "
        print " "
        print " "
        print " "
        print "##############################################################################################################"
        print "##############################      EXECUTANDO    PASSO 03      ##############################"
        print "##############################################################################################################"
```

```
        print "
        "
        print "   Verificando a presença de endereços IP em blacklists (span, hacker, vírus).
                "
        print "   Com essa informação um administrador de rede pode rapidamente
determinar se houve uma       "
        print "   violação de política ou intrusão em sua infraestrutura.
                "
        print "
        "
        print "   EXECUTANDO.... (AGUARDE)
                        "
        print "
        "
        print
"##############################################################################
######################"
        print
"##############################################################################
######################"
        print
"##############################################################################
######################"




def banner_A_1_1_8():
        print " "
        print " "
        print " "
        print " "
        print
"##############################################################################
######################"
        print "############################      EXECUTANDO    PASSO 04
##############################"
        print
"##############################################################################
######################"
        print "
        "
        print "   Utilizando motores de busca para coletar informações sobre o alvo,
protegendo              "
        print "   a anonimidade do pentester.
                "
```

```
        print "
        "
        print "   EXECUTANDO.... (AGUARDE)
                "
        print "
        "
        print
"##############################################################################
######################"
        print
"##############################################################################
######################"
        print
"##############################################################################
######################"

def banner_A_1_1_18():
        print " "
        print " "
        print " "
        print " "
        print
"##############################################################################
######################"
        print "##############################      EXECUTANDO    PASSO 05
##############################"
        print
"##############################################################################
######################"
        print "                                                                                "
        print "   Identificando contas de email válidas para tentar conectar ao servidor de
email alvo."
        print "                                                                                "

        print "   EXECUTANDO.... (AGUARDE)
                "
        print "                                                                                "

        print
"##############################################################################
######################"
        print
"##############################################################################
######################"
        print
"##############################################################################
######################"
```

```
def banner_A_1_1_23():
        print " "
        print " "
        print " "
        print " "
        print
"##############################################################################
#####################"
        print "############################      EXECUTANDO    PASSO 06
##############################"
        print
"##############################################################################
#####################"
        print "
        "
        print "   Servidores DNS relacionam endereços IP com o hostname que determinada
máquina possui. "
        print "   Essa informação fica arquivada na zona do DNS. Transferencia de zona entre
servidores "
        print "   é uma ação lícita adotada para sincronizar o banco de dados do servidor DNS
primário         "
        print "   com o secundário. Contudo, um atacante externo pode forjar uma solicitação
e requerer "
        print "   a zona de um servidor. Caso tenha sucesso, informações internas da rede
podem ser obtidas"
        print "   pelo atante, viabilizando o mapeamento de toda rede de uma organização.
                "
        print "
        "
        print "   Realizando tentativa de Transferência de Zona
                "
        print "
        "
        print "   EXECUTANDO.... (AGUARDE)
                "
        print "
        "
        print
"##############################################################################
#####################"
        print
"##############################################################################
#####################"
```

```
        print
"##############################################################################
######################"


def banner_A_1_1_24():
        print " "
        print " "
        print " "
        print " "
        print
"##############################################################################
######################"
        print "################################     EXECUTANDO    PASSO 07
##############################"
        print
"##############################################################################
######################"
        print "                                                                              "
        print "   Quando o servidor DNS está bem configurado e não é possível realizar a
Tranferência"
        print "   de Zona, existe outra maneira de mapear a rede alvo através de consultas
DNS.   "
        print "   Esse \"ataque\" consiste no envio de requisições DNS buscando nomes
comuns de      "
        print "   hostname (ex: router.alvo.com, firewall.alvo.com, etc)
        "
        print "                                                                              "
        print "   Com isso é possível descobrir o IP associado a cada hostname e mapear
grande parte "
        print "   da rede alvo.                                                              "
        print "                                                                              "

        print "   EXECUTANDO.... (AGUARDE)
                "
        print "                                                                              "

        print
"##############################################################################
######################"
        print
"##############################################################################
######################"
        print
"##############################################################################
######################"
```

```
def banner_A_1_1_26():
        print " "
        print " "
        print " "
        print " "
        print
"##############################################################################
######################"
        print "############################     EXECUTANDO    PASSO 08
##############################"
        print
"##############################################################################
######################"
        print "
        "
        print "   Clonando completamente o website do domínio alvo.
                "
        print "
        "
        print "   EXECUTANDO.... (AGUARDE)
                "
        print "
        "
        print
"##############################################################################
######################"
        print
"##############################################################################
######################"
        print
"##############################################################################
######################"


def banner_B_1_1_28():
        print " "
        print " "
        print " "
        print " "
        print
"##############################################################################
######################"
        print "############################     EXECUTANDO    PASSO 09
##############################"
```

```
        print "##############################################################################################################"
        print "
        "
        print "   Encontrar hosts ativos é o primeiro passo (ou um dos primeiros passos) no mapeamento   "
        print "   da rede. Esse passo pode reduzir severamente a quantidade de alvos a serem testados.         "
        print "   Geralmente essa ação utiliza o protocolo ICMP, mas algumas ferramentas utilizam TCP. "
        print "   Pode ser interessante que o administrador da rede bloqueie o protocolo ICMP.            "
        print "
        "
        print "   EXECUTANDO.... (AGUARDE)
                  "
        print "
        "
        print "##############################################################################################################"
        print "##############################################################################################################"
        print "##############################################################################################################"

def banner_B_1_1_29_B_2_1_1_e_B_2_1_1_2():
        print " "
        print " "
        print " "
        print " "
        print "##############################################################################################################"
        print "##############################     EXECUTANDO    PASSO 10     ##############################"
        print "##############################################################################################################"
        print "                                                                                                "
        print "   TCP e UDP port scan encontram todas as portas TCP e UDP \"listening\", \"closed\"       "
        print "   ou \"filtred\" em um determinado host. Com isso é possível descobrir quais portas  "
```

```
        print "   possuem serviços TCP e UDP ativos.
                "
        print "   Em seguida, para cada porta aberta, deve ser identificada a versão do serviço
que     "
        print "   está sendo executado, através da técninca conhecida como Banner Grabing.
                "
        print "                                                                                "
        print "   EXECUTANDO.... (AGUARDE)
                "
        print "                                                                                "

        print
"##############################################################################
######################"
        print
"##############################################################################
######################"
        print
"##############################################################################
######################"

def banner_B_2_1_10():
        print " "
        print " "
        print " "
        print " "
        print
"##############################################################################
######################"
        print "############################      EXECUTANDO    PASSO 11
##############################"
        print
"##############################################################################
######################"
        print "
        "
        print "   Utilizando informações obtidas durante o Banner Grabbing e realcionando
com outros     "
        print "   dados, como o fingerprint e scanning, a ação de enumeração permite inferir
novas  "
        print "   informações da rede alvo, bem como identificar falso positivos.
        "
        print "
        "
        print "   EXECUTANDO.... (AGUARDE)
                "
```

```
        print "
        "
        print
"##############################################################################
######################"
        print
"##############################################################################
######################"
        print
"##############################################################################
######################"


def banner_integraNessus():
        print " "
        print " "
        print " "
        print " "
        print
"##############################################################################
######################"
        print "##############################     EXECUTANDO    PASSO 12
##############################"
        print
"##############################################################################
######################"
        print "
        "
        print "   Utilizando a ferramenta Nessus para gerar um relatório de vulnerabilidades
do IP   "
        print "   alvo, complementando o relatório do Sapiens.
                "
        print "
        "
        print "   EXECUTANDO.... (AGUARDE)
                "
        print "
        "
        print
"##############################################################################
######################"
        print
"##############################################################################
######################"
        print
"##############################################################################
######################"
```

```
inicio()
geraHTML()
A_1_1_2()
A_1_1_3()
A_1_1_5()
A_1_1_8()
A_1_1_18()
A_1_1_23()
A_1_1_24()
A_1_1_26()
B_1_1_28()
B_1_1_29_B_2_1_1_e_B_2_1_1_2()
integraNessus()
comandoUsuario()



# fechamento das tag html e geração da mensagem de rodapé da página html
def fechaHTML():

	# Cria a variável "horaInicio" cujo conteúdo será o dia e a hora de início do uso da ferrametna "sapiens".
	horaTermino = commands.getoutput('date')
	comando('echo "O pentest foi iniciado às \'%s\' e concluído às \'%s\'" >> %s' %(horaInicio, horaTermino, relatorio))

	pagina=open("/var/www/html/sapiens.html","a")
	pagina.write("""
</div>
		<div id="footer">
			<p>
				Webpage made by <a href="/" target="_blank">[your name]</a>
			</p>
		</div>
	</div>
</body>
</html>
	""")
	pagina.close()

fechaHTML()
```